Distributed Source Coding: Tools and application to video compression

Velotiaray Toto-Zarasoa

N° d'ordre : 4221 ANNÉE 2010

UNIVERSITÉ DE RENNES 1

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : TRAITEMENT DU SIGNAL

Ecole doctorale MATISSE

présentée par

Velotiaray TOTO-ZARASOA

préparée à l'**INRIA** Rennes-Bretagne Atlantique
(**I**nstitut **N**ational de **R**echerche en **I**nformatique et **A**utomatique, Équipe **TEMICS**)

Composante universitaire : **IFSIC**

Codage de sources distribuées : Outils et Applications à la compression Vidéo.

Thèse soutenue à Rennes le 29 novembre 2010

devant le jury composé de :

Frédéric DUFAUX
Directeur de Recherche CNRS, Télécom ParisTech / Rapporteur

Michel KIEFFER
Maître de Conférences, Université de Paris Sud / Rapporteur

Enrico MAGLI
Assistant Professor, Politecnico di Torino / Éxaminateur

Pierre SIOHAN
Ingénieur, Orange Labs / Éxaminateur

Christine GUILLEMOT
INRIA / Directeur de thèse

Aline ROUMY
INRIA / Co-directeur de thèse

Contents

Introduction

La compression numérique est aujourd'hui au cœur de notre quotidien. En effet, il est rare que les données brutes (format "RAW") relevées par les capteurs numériques soient conservées telles quelles. Ces formats RAW ne sont directement utilisés que par les professionnels du numérique, afin de travailler sur une représentation la plus fidèle possible des données. Pour le plus grand nombre d'utilisateurs finaux, les images sont par exemple conservées sous formats JPEG, JPEG2000, PNG, GIF, etc. permettant d'économiser la place disponible sur le support de stockage. Les fichiers audio sont compressés sous formats MP3, OGG, WMA, etc. Enfin les fichiers vidéo, les plus gros en termes d'espace occupé, sont sauvegardés sous formats MPEG, MP4, AVI, etc. Dans cette thèse, nous nous intéressons à une compression particulière de la vidéo, qui favorise une faible complexité d'encodage, en supposant que le décodeur dispose d'assez de ressources pour effectuer les opérations les plus gourmandes en énergie : le *Codage Vidéo Distribué*.

Une séquence vidéo peut être vue comme une succession d'images, qui, placées les unes après les autres avec un intervalle de temps assez court, donnent l'impression de mouvement. Le but recherché dans la compression de la vidéo est de minimiser la taille finale de la vidéo tout en assurant un confort visuel acceptable ; ceci se fait en exploitant la corrélation qui existe dans les images qui se suivent. En effet, les pixels adjacents d'une vidéo, aussi bien dans le temps (entre deux images différentes) que dans l'espace (entre deux pixels proches dans une même image), se ressemblent fortement. Il semble donc intelligent, étant donné la connaissance d'un pixel donné, de ne transmettre au décodeur que la différence avec les pixels alentours. Par exemple, l'exploitation de la corrélation spatiale (on parle de codage *intra*) est implémentée, dans le codage MPEG, par la mise en œuvre d'une transformation DCT (Discrete Cosine Transform) des images à coder. L'exploitation de la corrélation temporelle (on parle de codage *inter*) est mise en œuvre par exemple par un module de prédiction temporelle compensée en mouvement. La mise en œuvre des ces outils nécessitent à l'encodeur un grand apport d'énergie pour la compression vidéo traditionnelle, or, dans les terminaux modernes (téléphones mobiles dotés d'une caméra, réseaux de capteurs, etc.) l'appareil énergétiquement autonome ne dispose que de peu de ressources pour mener à bien la compression. Il en résulte une qualité amoindrie de la vidéo obtenue. Le but de la compression vidéo distribuée est de migrer cette complexité vers le décodeur en trouvant un compromis entre les besoins énergétiques de chacun, l'encodeur ne prenant en charge que des opérations basiques. Ceci est rendu possible par un concept apparu en 1973 : le *Codage de Sources Distribuées*.

Le codage de sources distribuées [SW73] vise à compresser une ou plusieurs

sources discrètes *corrélées*. Le terme "distribué" signifie que les encodeurs des sources ne peuvent pas communiquer entre elles, ce qui implique que leur encodage est *disjoint*. Si leur décodage s'effectue néanmoins *conjointement*, le résultat de Slepian et Wolf [SW73] stipule que les performances atteignables sont similaires à celles atteignables par un encodage conjoint. Plus précisément, on sait depuis le Théorème de Shannon [Sha59] que le débit minimal permettant de retrouver sans erreur deux sources corrélées, que l'on va noter X et Y, est leur entropie conjointe $H(X,Y)$; seulement, le Théorème de Shannon n'est valable que si l'encodage des deux sources se fait conjointement. Moyennant l'envoie d'une information minimale pour chaque source, s'élevant à son entropie conditionnelle ($H(X|Y)$ pour X et $H(Y|X)$ pour Y), Slepian et Wolf ont découvert que la somme minimale des débits des deux sources peut rester à l'entropie conjointe pour retrouver les deux sources sans erreur.

Lorsque l'une des deux sources, mettons Y, est disponible au décodeur (en la compressant à son entropie $H(Y)$ par exemple), on parle de codage *asymétrique* de Slepian-Wolf (SW); dès lors, Y est considérée comme une *information adjacente* pour le décodage de X. L'autre source, X, peut être compressée à son entropie conditionnelle $H(X|Y)$ pour pouvoir être décodée sans erreur. Une manière efficace d'effectuer le codage distribué est l'utilisation de codes canal [Wyn74], car chacune des sources peut être vue comme une version bruitée de l'autre ; dès lors une interprétation du décodage d'une source est de "corriger les erreurs" présents dans l'autre source. Un codage canal peut alors être mis en œuvre pour le codage de SW; si le code canal utilisé atteint la capacité du canal de corrélation, alors il atteint aussi la borne de SW. Le résultat asymétrique de Slepian et Wolf à été généralisé par Wyner et Ziv [WZ76] pour le codage de sources à valeurs réelles. Le codage asymétrique est un cas particulier du codage *non-asymétrique* de SW, où les deux sources sont compressées à des débits se situant entre leurs entropies et leurs entropies conditionnelles.

Dans cette Thèse, nous nous intéressons à ces deux cas *asymétrique* et *non-asymétrique*, en proposant des modèles de sources binaires mieux adaptés aux sources rencontrées dans les applications pratiques ainsi que des modèles de corrélations également plus fidèles aux applications pratiques. Ces développements sont motivés par les nombreuses applications du codage asymétrique (par exemple pour l'authentification de fichiers d'images et d'audios [LVG07b, LVG07a], et, une application plus particulièrement intéressante pour cette Thèse, le codage vidéo distribué [PR02, AZG02, AAD+07, dis]) et non-asymétrique (par exemple pour les réseaux de capteurs où une optimisation particulière des débits montants doit être mise en place [RG07], ou l'application au codage des "light fields" (champs de lumière) [GD05]). Nous proposons également des outils de décodages adaptés à ces modèles de sources et canaux de corrélation ainsi que des outils d'estimation de leurs paramètres, afin d'atteindre les bornes de Slepian-Wolf correspondantes. Les modèles et outils que nous avons développés pour le codage asymétrique sont ensuite investis dans la compression de vidéos pour améliorer la *performance débit-distorsion* d'un codec vidéo distribué existant. Ce mémoire de Thèse est structuré en deux parties regroupant respectivement l'état de l'art du domaine du codage de sources distribuées (Chapitre 1) et celui du codage vidéo distribuée (Chapitre 2) ; nos contributions pour l'étude

de ces domaines sont présentées dans les Chapitres 3 à 5.

Le premier Chapitre est dédié à la présentation du codage de sources distribuées, en rappelant les outils existant dans la littérature pour atteindre les bornes de Slepian-Wolf (SW). Une présentation des résultats liés au codage de Wyner-Ziv (WZ) est également faite. Les deux outils sur lesquels nous nous baserons dans nos contributions (les codages Turbo [RLG07] et LDPC - Low-Density Parity-Check - [LXG02] basés syndromes) sont décrits plus en détails dans leurs encodages et leurs décodages. Enfin, deux modèles de canaux sont présentés pour modéliser la corrélation entre les sources distribuées : le *canal binaire symétrique sans mémoire* et le *canal avec mémoire de Gilbert-Elliott* (GE) [Gil60, Ell63].

Le Chapitre 2 est l'occasion de revoir les bases du codage vidéo distribué. Nous y rappelons les motivations qui ont incité sa création en 2002 [PR02, AZG02], et nous présentons le codec vidéo distribué [AAD+07, dis] sur lequel se basent nos contributions expérimentales : le codec développé sous le projet Européen DISCOVER (Distributed Coding for Video Services, [dis]), qui est l'un des codecs vidéo distribués les plus aboutis du moment. D'une manière générale, un codec vidéo distribué vise à diminuer de façon significative la complexité de l'encodeur, par rapport aux codecs classiques MPEG, H264 dont l'encodage est extrêmement complexe. Dans le codec du projet DISCOVER, les images sont structurées par groupes d'images, dans lesquels les images clés sont codées en mode intra, et les images intermédiaires (appelées images de WZ) sont codées selon le principe de WZ. Plus précisément, les images de WZ sont transformées (par une DCT - Discrete Cosine Transform -) puis quantifiées, et les coefficients issus de la quantification sont réduits à l'état binaire, pour former des *plans de bits*. Chaque plan de bits obtenu est finalement encodé par un code canal pour générer des *syndromes*. Le décodeur a un rôle plus complexe. En effet, il a pour charge de construire une image information adjacente pour chaque image de WZ, via une interpolation compensée en mouvement des images clés et celles précédemment décodées. Cette image adjacente subit ensuite le même traitement que les images de WZ, pour en obtenir des informations adjacentes pertinentes pour le décodage des plans de bits des images de WZ. Le décodage des plans de bits de WZ se fait à l'aide d'un décodeur adaptatif en débit [VAG06], et chaque requête de bits de syndrome additionnel s'effectue par une voie de retour.

Dans le Chapitre 3, nous exposons les modèles de sources binaires *non-uniforme* et avec mémoire (*source de GE*, nommé ainsi d'après sa similitude avec le modèle de canal avec mémoire de GE), qui assurent chacune des bornes de compression plus avantageuses que le modèle de source binaire uniforme (classiquement utilisé dans la littérature), pour le codage de SW. Des méthodes d'estimation de leurs paramètres sont décrites. Pour modéliser la corrélation entre ces sources, nous présentons deux modèles innovants de canaux sans mémoire : les canaux binaires symétriques *additif* et *prédictif*, qui impliquent des bornes très différentes pour le codage de SW. Nous terminons ce Chapitre par la description d'une nouvelle méthode d'estimation hors ligne du paramètre du canal binaire symétrique, habituellement noté p dans la littérature, qui atteint la borne de Cramer-Rao lorsque le code est suffisamment puissant pour estimer les sources quasiment sans erreur.

Faisant écho au Chapitre 3, le Chapitre 4 regroupe les outils que nous avons créés, basés sur les codes LDPC et Turbo, pour effectuer des codages de SW *asymétrique*

et *non-asymétrique* efficaces pour les modèles de sources présentés au Chapitre 3. Des algorithmes originaux pour les décodages des sources binaires non-uniforme ou avec mémoire sont présentés, prenant la forme d'algorithmes EM (Expectation-Maximization, [Rab89]) adaptés au problème *asymétrique* de SW pour le décodage et l'estimation conjoints des sources et de leurs paramètres. Par la suite, nous présentons les outils adaptés au codage *non-asymétrique* de SW, à savoir des codes Turbo et LDPC robustes et adaptatifs en débit ; nous dérivons des conditions nécessaires et suffisantes, sur les propriétés des codes utilisés, pour décoder efficacement, et sans propagation d'erreurs, des sources uniformes ou non-uniformes.

L'expérimentation des modèles et des outils décrits dans les Chapitres 1 et 2 pour le codage vidéo distribué est présenté dans le Chapitre 5. Nous y démontrons la pertinence des deux modèles de sources binaires (non-uniforme et de GE), et des deux modèles de canal de corrélation (canal binaire symétrique additif et prédictif), pour les modélisations, respectivement, des distributions des plans de bits générés par le codec vidéo distribué, et de la corrélation entre ces plans de bits et l'information adjacente extraite par le décodeur. Nous démontrons que des gains considérables par rapport au codec initial, en termes de performance débit-distorsion du codec obtenu, peuvent être observés ; ces gains peuvent aller jusqu'à 5.7% pour une modélisation des plans de bits comme sources non-uniformes, et jusqu'à 10.14% pour une modélisation des plans de bits comme sources de GE.

Résumé

1 - Le codage de sources distribuées

Le codage de sources distribuées est l'art de compresser séparément des sources corrélées, et de les décompresser conjointement au décodeur. Ce concept a été introduit par Slepian et Wolf en 1973 [SW73] dans le cas de sources discrètes. Slepian et Wolf ont démontré qu'il n'y a qu'une légère dégradation à encoder les sources séparément, par rapport au cas où elles sont encodées conjointement. Dans cette thèse, nous nous intéressons essentiellement au codage distribué de deux sources binaires corrélées, notées X et Y. Notons d'autre part R_X et R_Y leurs débits respectifs. Si un encodage et un décodage conjoints permet de borner la somme $R_X + R_Y$ à l'entropie conjointe $H(X,Y)$ pour retrouver les deux sources sans erreurs (résultat de Shannon en 1959), ce résultat est conditionnellement vrai pour le codage de Slepian-Wolf (SW), car il faut en plus vérifier que $R_X \geq H(X|Y)$ et $R_Y \geq H(Y|X)$. La dégradation provient donc de l'obligation à envoyer au minimum ces informations équivalentes aux entropies conditionnelles pour les deux sources. Ce résultat est résumé sur la Fig. 1 ci-dessous montrant la région et la borne des débits atteignables pour le codage de SW :

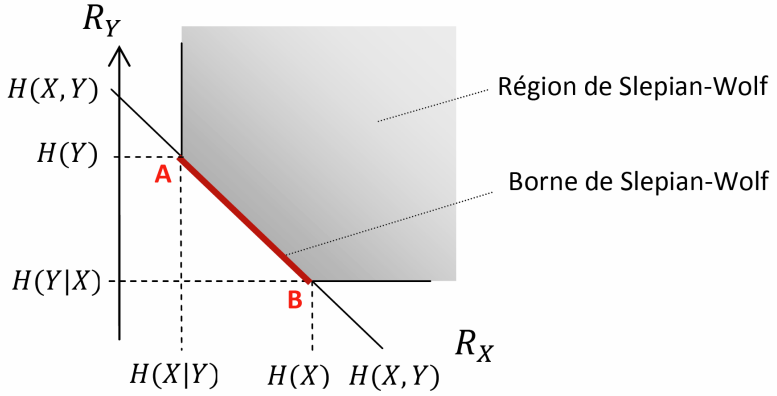

Figure 1: La région des débits de SW.

1.1 - Codage distribué asymétrique

Un cas particulier du codage de SW consiste à compresser une source X à son entropie conditionnelle, et de la décompresser connaissant la source Y au décodeur;

c'est le codage dit *asymétrique* de SW (point A sur la Fig. 1), aussi appelé "codage distribué avec information adjacente au décodeur". La source Y peut alors être perçue comme une version bruitée de X, et retrouver X consiste à "corriger les erreurs" présents dans Y. C'est pourquoi [Wyn74] démontre qu'une façon optimale d'atteindre le point A (resp. B) est l'emploi de codes canal pour la compression de X (resp. Y). En pratique, ceci est efficacement effectué par les codes LDPC (Low-density Parity-Check) [Gal62] ou Turbo [BG96] par exemple.

La généralisation de ce Théorème de SW pour la compression avec pertes se retrouve dans le Théorème de Wyner-Ziv (WZ) [WZ76]. On ne parle alors plus de "région des débits", mais de "région des débits-distorsions", dû à la reconstruction intrinsèquement imparfaite. Plus précisément, si l'on appelle D la distorsion souhaitée à la reconstruction de X; $\hat{X}$ la version reconstruite de X; $d\left(X,\hat{X}\right)$ la mesure de la distorsion entre X et $\hat{X}$; et $I\left(X;\hat{X}\right)$ l'information mutuelle entre X et $\hat{X}$, la compression avec pertes avec information adjacente disponible seulement au décodeur est régie par la fonction suivante [WZ76] :

$$R^{\star}_{X|Y}(D) = \min_{p\left(Z,Y,\hat{X}\right)\in\mathbb{M}(D)} \left[I(X;Z) - I(Y;Z)\right] \tag{1}$$

où $\mathbb{M}(D)$ est l'ensemble des distributions $p\left(x,y,z\right)$ possibles, telles que $\exists f : Y\times Z \rightarrow \hat{X} : \mathbb{E}_{p(X,Y,Z)}\left[d\left(X,\hat{X}\right)\right] \leq D$, et où $\hat{X} = f(Y,Z)$.

Cette fonction débits-distorsions est à comparer au cas où Y est connue également à l'encodeur ; [Ber72] donne :

$$R_{X|Y}(D) = \min_{p\left(X,Y,\hat{X}\right):\mathbb{E}_{p(X,Y,\hat{X})}\left[d\left(X,\hat{X}\right)\right]\leq D} I\left(X;\hat{X}|Y\right) \tag{2}$$

Il est démontré dans [WZ76] qu'en règle générale, $R_{X|Y}(D) \leq R^{\star}_{X|Y}(D)$. Par contre, lorsque les sources X et Y suivent des distributions normales, Wyner démontre dans [Wyn78] que $R_{X|Y}(D) = R^{\star}_{X|Y}(D)$. Cette égalité est démontrée en 2003 par Pradhan [PCR03] pour des sources quelconques dont la corrélation peut se modéliser par une loi normale.

1.2 - Codage non-asymétrique de SW

L'autre aspect du codage de sources distribuées est de travailler à tout point entre A et B (voir Fig. 1), ce qui correspond au cas non-asymétrique du codage de SW. Ceci peut se révéler particulièrement intéressant pour résoudre le problème d'allocation des ressources entre les différents nœuds d'un réseau de capteurs [RG07], afin de capturer plus efficacement les différences de conditions de transmissions des données entre chaque capteur et le récepteur unique. Un moyen d'atteindre ces points est également de se baser sur l'utilisation de codes canal, comme démontré dans [GD05, TL05b].

2 - Contributions pour l'estimation du paramètre du canal binaire symétrique

Notons que pour les codages asymétrique et non-symétrique de SW, il est généralement adopté dans la littérature de considérer le paramètre de la corrélation connu, lorsque cette dernière est modélisée comme un *canal binaire symétrique* virtuel. Nous proposons une méthode rapide et efficace pour estimer ce paramètre, en assurant d'atteindre la borne de Cramer-Rao lorsque le code utilisé est assez puissant pour retrouver les sources sans erreur. Plus précisément, notre méthode repose sur l'existence d'une bijection entre la valeur du paramètre du canal (usuellement noté "p") et la distribution de la différence entre les syndromes des sources corrélées. Une inversion de cette fonction bijective permet de retrouver une estimée $\hat{p}$ dont le biais diminue avec la longueur du code, et dont le MSE (Mean Square Error) est proche de la borne de Cramer-Rao. Cette estimée est ensuite raffinée grâce à un algorithme EM, pour atteindre la borne de Cramer-Rao pour les valeurs de p permettant de retrouver X et Y sans trop grand distorsion par le décodage canal.

3 - Contributions pour le codage non-asymétrique de SW

Ici, nous nous situons dans la configuration où les deux sources sont compressées afin de travailler à n'importe quel point du segment entre les points A et B (voir Fig. 1), pour n'importe quel niveau de corrélation p entre les deux sources corrélées. Tout d'abord, nous nous attaquons au problème du codage de SW *non-asymétrique et adaptatif en débit* ; nous proposons alors un unique codec, basé sur le codage LDPC asymétrique et adaptatif en débit introduit dans [VAG06] et sur un codage LDPC non-asymétrique proposé dans [GD05]. Le code unique résultant est efficace, mais lorsqu'il est réalisé en deux étapes, ce décodage est sujet à une *propagation d'erreurs*. Afin de réduire cette propagation d'erreurs, nous cherchons des solutions basés sur les codes Turbo, et nous aboutissons à des conditions nécessaires et suffisantes pour rendre robuste le codage non-asymétrique. Deux solutions d'optimisation des codes sont proposées pour ce problème :

- le premier consiste à transmettre des informations supplémentaires quant aux états du treillis du code Turbo lors de l'encodage des deux sources,
- le deuxième consiste à choisir un code Turbo dont la représentation matricielle est bloc diagonale.

Enfin, nous terminons cette étude du codage non-asymétrique par la recherche et l'élaboration d'un code LDPC effectuant le codage non-asymétrique de sources binaires non-uniformes. Nous montrons que la matrice correspondant au code LDPC doit avoir une sous-partie triangulaire ; le décodage doit aussi être modifié pour prendre en compte la non-uniformité de chacune des sources, ainsi que le modèle de corrélation (additif / prédictif) entre chaque source.

4 - Codage vidéo distribué

Une application prometteuse du codage *asymétrique* de SW réside dans le codage vidéo distribué. Le principe est d'exploiter efficacement la corrélation entre les images successives de la vidéo par un système de codage de sources distribuées. Le codage vidéo distribué diffère des codages conventionnels (MPEG, H264) par sa complexité d'encodage très réduite, ce qui lui permet d'améliorer par exemple la durée de vie de l'alimentation d'une caméra embarquée disposant de ressources énergétiques restreintes. En contrepartie, la complexité du décodeur est plus grande par rapport à celle des décodeurs conventionnels.

Le tout premier dispositif pratique pour le codage vidéo distribué a été proposée par Puri et Ramchandran [PR02] à l'université de Berkeley, en 2002, sous la dénomination "PRISM". Pendant la même période, une autre solution de codage vidéo distribué a été conçue par Girod et son équipe [AZG02], à l'université de Stanford. Une troisième solution a vu le jour en 2005, développée dans le cadre du projet Européen DISCOVER [AAD+07, dis], basée sur l'architecture de la solution provenant de Stanford. Ce dernier codec est aujourd'hui l'un des plus avancés dans ce domaine. Voici une description rapide des deux composantes du codec :

L'*encodeur* structure les images successives par groupes d'images, dans lesquels les images clés sont codées en mode *intra*, et les images intermédiaires (aussi appelées images de WZ) sont codées selon un *principe distribué*. Plus précisément, les images de WZ sont transformées (par DCT - Discrete Cosine Transform -) puis quantifiées, et les coefficients issus de la quantification sont réduits à l'état binaire, pour former des *plans de bits*. Chaque plan de bits obtenu est finalement encodé par un code canal (un code LDPC basé syndrome dans notre cas) pour en obtenir des *syndromes* pouvant être stockés dans un *buffer*, pour une transmission graduelle vers le décodeur.

Le *décodeur* a un rôle plus complexe. En effet, il a pour charge de construire une information adjacente pour chaque image de WZ, via une interpolation compensée en mouvement des images actuellement disponibles qui lui sont les plus proches. L'*image adjacente* obtenue subit ensuite le même traitement que les images de WZ, pour en obtenir des informations adjacentes exploitables pour le décodage LDPC des plans de bits des images de WZ. Dans l'état actuel du codec, la corrélation, entre les plans de bits de WZ et l'information adjacente leur correspondant, est modélisée par un canal Laplacien. Le décodage des plans de bits de WZ se fait à l'aide du décodeur adaptatif en débit décrit dans [VAG06], et chaque requête de bits de syndrome additionnel s'effectue par une voie de retour vers le buffer.

Si nous nous intéressons à la modélisation des plans de bits extraits de la vidéo au niveau de l'encodeur du codec DISCOVER, leur distribution est classiquement supposée uniforme, c'est-à-dire $\mathbb{P}(1) = \mathbb{P}(0) = 0.5$. Or notre analyse approfondie démontre qu'ils sont plus fidèlement représentés par un modèle de source binaire *non-uniforme*, c'est-à-dire $\mathbb{P}(1) = p_X, \mathbb{P}(0) = (1 - p_X)$, où $p_X \neq 0.5$. De ce constat, nous nous sommes penchés sur la production d'un codeur LDPC adapté à cette non-uniformité, pour bénéficier d'un débit plus réduit par rapport au codage d'une source uniforme, et ainsi améliorer la caractéristique débit-distorsion de la vidéo après son décodage. Une analyse plus poussée des plans de bits démontre que la distribution

binaire de ces mêmes plans de bits est encore plus fidèlement modélisable par un modèle de source avec mémoire inspiré du modèle de canal de Gilbert-Elliott (GE) [Gil60, Ell63], que nous appelons *source de GE* ; une autre de nos productions est aussi la construction d'un codeur LDPC capable d'exploiter la mémoire dans ces sources, pour améliorer davantage la caractéristique débit-distorsion du codec vidéo distribué. La Section qui suit décrit l'exploitation de cette mémoire à l'aide d'un décodeur LDPC.

5 - Le modèle de source de Gilbert-Elliott (GE)

5.1 - Présentation du modèle

Notons X la source de GE, et $\mathbf{x}$ sa réalisation de longueur N. Notons par ailleurs Σ le processus Markovien de ses états, et σ sa réalisation de longueur N. La Fig. 2 décrit la méthode de génération d'une source de GE, où le paramètre du modèle est noté $\theta_X = \{p_{\mathbf{0}}, p_{\mathbf{1}}, t_{\mathbf{10}}, t_{\mathbf{0,1}}\}$. Deux états gouvernent la génération des symboles de la source: $\mathbf{0}$ et $\mathbf{1}$ dans lesquels, à chaque position $n \in [1, N]$, la source suit des lois de Bernoulli de paramètres respectifs $p_{\mathbf{0}} = \mathbb{P}_{\theta_X}(X_n = 1|\Sigma_n = \mathbf{0})$ et $p_{\mathbf{1}} = \mathbb{P}_{\theta_X}(X_n = 1|\Sigma_n = \mathbf{1})$, telles que $p_{\mathbf{0}}, p_{\mathbf{1}} \in [0, 1]$ et $p_{\mathbf{0}} \leq p_{\mathbf{1}}$. Le basculement d'un état à l'autre est régi par les paramètres de transitions $t_{\mathbf{01}} = \mathbb{P}_{\theta_X}(\Sigma_n = \mathbf{1}|\Sigma_{n-1} = \mathbf{0})$ et $t_{\mathbf{10}} = \mathbb{P}_{\theta_X}(\Sigma_n = \mathbf{0}|\Sigma_{n-1} = \mathbf{1}) \in [0, 1]$. La source contient donc une mémoire *infinie*, qui est plus persistante lorsque les paramètres de transitions sont plus faibles. Nous regroupons les paramètres de la source X dans le paramètre unique $\theta_X = (p_{\mathbf{0}}, p_{\mathbf{1}}, t_{\mathbf{10}}, t_{\mathbf{01}})$. Il est montré dans [Cov75] que la source X de GE a une entropie-rate $H(\mathcal{X})$ plus petite que l'entropie d'une source uniforme. Cette entropie est efficacement calculable par la méthode présentée dans [Rez05].

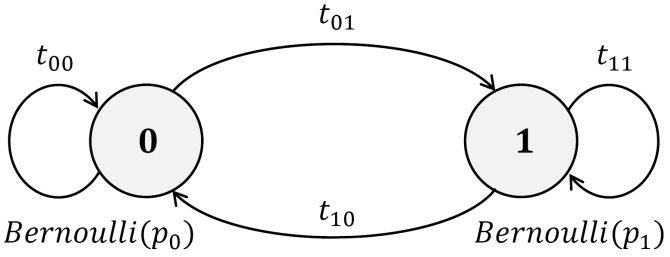

Figure 2: Diagramme décrivant la source de GE.

Venons-en au codage distribué asymétrique des sources de GE. Notons Z la différence entre X et Y, un canal binaire symétrique *additif* (c'est-à-dire $Y = X \oplus Z$) virtuel de paramètre p. La borne de SW pour la compression de X est ainsi donnée par :

$$H(\mathcal{X}|\mathcal{Y}) = H(p) - [H(\mathcal{Y}) - H(\mathcal{X})] \tag{3}$$

Cette entropie conditionnelle est plus petite de $H(\mathcal{Y}) - H(\mathcal{X})$ par rapport à l'entropie conditionnelle $H(X|Y) = H(p)$ d'une source uniforme. Cependant, il est

primordial de noter que cette Equation (3) n'est valable que pour le modèle de canal *additif* ; en effet, pour le modèle de canal *prédictif* (où $X = Y \oplus Z$), il n'y a aucun gain pour la compression de X par rapport à la compression classique d'une source uniforme. Le décodeur LDPC exploitant la mémoire dans la source X devra donc être différent selon les deux modèles de corrélation.

5.2 - Outil de décodage et d'estimation conjoints

Le décodeur que nous introduisons ici, qui permet d'exploiter la mémoire infinie de la source X, et d'estimer ses paramètres simultanément, est basé sur l'algorithme EM (Expectation-Maximization) [Rab89]. Ce dernier permet de mettre à jour les paramètres de la source au fur et à mesure de l'avancement du décodage ; la convergence de l'algorithme est donné par le résultat dans [Wu83]. Plus précisément, à chaque itération l, l'algorithme EM met à jour le paramètre θ_X, en maximisant la fonction de vraisemblance logarithmique moyenne donnée dans l'Equation (4), étant donnés la réalisation de Y, noté $\mathbf{y}$, le syndrome de X, noté $\mathbf{s_x}$, et l'estimée actuelle de θ_X, noté θ_X^l :

$$\theta_X^{(l+1)} = \arg\max_{\theta_X} \left(\mathbb{E}_{\mathbf{X},\mathbf{\Sigma_x}|\mathbf{Y},\mathbf{S_X},\theta_X^l} \left[\log \left(\mathbb{P}_{\theta_X}(\mathbf{y}, \mathbf{x}, \sigma_\mathbf{x}, \mathbf{s_x}) \right) \right] \right) \tag{4}$$

L'étape "E" (Estimation) de l'algorithme EM consiste à calculer cette fonction, à chercher une estimée des états σ par un *algorithme forward-backward*, et à chercher une estimée de $\mathbf{x}$ par un *décodage LDPC* modifié. L'étape "M" consiste à mettre à jour le paramètre θ_X en maximisant la fonction (4). Après chaque itération, θ_X^{l+1} est plus proche de θ_X que θ_X^l. Des estimées de $\mathbf{x}$ et de σ sont fournies en tant que produits dérivés de l'algorithme EM. En ce qui concerne le décodage LDPC, dans le cas d'un modèle de corrélation *additif*, des messages additionnels proviennent et partent des nœuds de variables vers les états, par rapport au décodage classique [LXG02] ; si le modèle de corrélation est prédictif, l'algorithme traditionnel est déjà le mieux que l'on puisse faire.

L'algorithme EM est initialisé avec les paramètres du modèle de GE provenant de l'information adjacente. En effet, comme X et Y sont des sources corrélées, les états de Y sont les mêmes que ceux de X, donc le paramètre θ_Y est la meilleure valeur pour θ_X^0 à ce niveau de l'algorithme. Pour trouver les paramètres de la source Y, un algorithme de Baum-Welch est employé ; ce dernier est initialisé avec les valeurs figées $\theta^0 = \{p_\mathbf{0}^0 = 0.49, p_\mathbf{1}^0 = 0.51, t_\mathbf{10}^0 = 0.1, t_\mathbf{01}^0 = 0.1\}$.

L'algorithme EM peut être arrêté après chaque itération si le vecteur décodé $\hat{\mathbf{x}}$ vérifie les équations de la matrice de parité, ou si un nombre maximal d'itérations est atteint (dans nos tests, 100 itérations sont suffisants pour obtenir une estimée acceptable de $\mathbf{x}$).

Notons qu'une source non-uniforme ou une source uniforme sont des cas particuliers de la source de GE, donc le décodeur est commun à ces trois modèles de sources du moment que la corrélation est modélisée par un *canal binaire symétrique*.

Nous sommes maintenant prêts à mener les tests de validation sur le codec DISCOVER, pour évaluer l'efficacité de notre estimateur-décodeur basé sur l'algorithme EM.

6 - Expérimentations sur le codec DISCOVER

6.1 - Validation des modèles proposés

Avant de remplacer l'ancien décodeur de SW de DISCOVER par le nouveau, basé sur l'algorithme EM, il nous faut vérifier la pertinence des deux modèles de sources que nous avons proposés. Pour ce faire, nous extrayons les plans de bits générés par le codec, et nous les analysons hors ligne afin de vérifier leurs distributions binaires.

Pour le modèle de source binaire non-uniforme, il faut calculer, pour chaque plan de bit, le pourcentage de "1" contenu dans le plan de bit. Cela nous fournit le paramètre p_X du plan de bit. Si p_X est différent de 0.5, alors le plan de bit est effectivement non-uniforme. Les résultats obtenus hors ligne sont concluants car la plupart des plans de bits sont hautement non-uniformes.

Pour le modèle de source de GE, il nous faut analyser les distributions de traînées de "1" pour les plans de bits. En effet, pour une source sans mémoire (c'est-à-dire uniforme ou non-uniforme), la probabilité $\mathcal{P}_k$ d'observer une traînée de longueur k est donnée par la formule $\mathcal{P}_k = (1 - p_X)p_X^{k-1}$. Il en ressort que le logarithme de la distribution des traînées de "1" est linéaire avec la longueur k de la traînée, ce qui n'est pas le cas pour une source de GE. Lors des analyses sur les plans de bits de la vidéo, cette vérification est aussi concluante : un nombre important de plan de bits possède effectivement de la mémoire.

Pour l'estimation du modèle de canal de corrélation, il ne nous est malheureusement pas possible d'estimer si le modèle de corrélation est *additif* ou *prédictif* ; aucun critère fiable ne permet à ce jour de faire la discrimination par l'observation des informations à notre disposition. Donc, notre solution est de tester les deux modèles et de choisir celui qui fournit un décodage plus rapide du code LDPC adaptatif en débit. Notons que ce n'est pas un système "aidé" car aucun élément additionnel n'est donné par rapport au système standard.

6.2 - Mise en œuvre du nouveau décodeur de SW dans DISCOVER

Une fois que les modèles de sources ont été définis et validés, et que les outils nécessaires à leurs décodages ont été élaborés, il nous reste à implémenter le tout à la place du décodeur de SW dans le codec DISCOVER. Nous observons ensuite les performances débits-distorsions du codec obtenu pour les cinq séquences vidéo *Hall Monitor*, *Foreman*, *Coastguard*, *Flower*, et *Soccer*. Nous les comparons enfin aux performances du codec standard. Les résultats sont en faveur du système que nous proposons : le gain en débit, pour la même qualité de reconstruction des vidéos, atteint 5.7% pour la séquence *Soccer*, au plus fort PSNR, lorsque les plans de bits sont modélisés comme des sources non-uniformes ; ce gain atteint 10.14%, lorsque les plans de bits sont modélisés comme des sources de GE. Les résultats sont résumés dans le Tableau 1 pour la modélisation avec des sources non-uniformes, et dans le tableau 2 pour la modélisation avec des sources de GE :

Ces résultats confirment que les modélisations des plans de bits comme des réalisations de *sources non-uniformes* ou de *sources de GE* sont plus efficaces que la modélisation en source uniforme, pour le codage vidéo distribué. D'autre part, il y a environ deux fois plus de gain avec le modèle de GE qu'avec le modèle non-uniforme.

Séquence	**Gain en débit (kbps)**	**Gain en débit (%)**
Hall Monitor	0.94	1
Foreman	5.88	2.68
Coastguard	2.04	1.15
Flower	2.97	1.47
Soccer	16.57	5.7

Table 1: Gains lorsque les plans de bits sont modélisés en sources non-uniformes.

Séquence	**Gain en débit (kbps)**	**Gain en débit (%)**
Hall Monitor	2.54	2.73
Foreman	8.76	4
Coastguard	4.28	2.41
Flower	5.96	2.95
Soccer	29.55	10.14

Table 2: Gains lorsque les plans de bits sont modélisés en sources de GE.

Néanmoins, le choix des sources non-uniformes est motivé par la moindre complexité du décodage, par rapport aux sources avec mémoire.

Part I

State of the art

Chapter 1

Distributed source coding

This chapter presents the theoretical background of Distributed Source Coding (DSC), and introduces some existing practical solutions. A quick overview of the different frameworks based on channel codes is depicted. Then, the practical implementations of two DSC codes, namely syndrome-based Turbo and Low-Density Parity-Check (LDPC) codes, are detailed. This choice of codes is motivated by the reputation of Turbo and LDPC codes to be *near capacity-achieving* codes. As the DSC problem can be formally interpreted as a channel coding one, these two codes are also near optimal for *DSC*. The results on discrete binary sources are first presented in Section 1.2 in which the correlation between the sources is modeled as a Binary Symmetric Channel (BSC). In Section 1.2.6, we review distributed coding using source codes, which is shown to exhibit performances that are comparable to that of Turbo codes in Section 4.1.1.4. Then, in Section 1.4.1.2, the background on the BSC parameter estimation is reviewed. Afterward, the generalization to continuous sources is described in Section 1.5. We end this chapter with an overview of our contributions for DSC in Section 1.6.

1.1 A word on notation

In this Thesis, uppercase variables (X, Y, Z, Σ) refer to *random stochastic processes*, uppercase variables $(X_n, Y_n, Z_n, \Sigma_n)$ refer to *random variables* at instant n, and lowercase variables $(x_n, y_n, z_n, \sigma_n)$ refer to their respective *realizations*. Bold uppercase variables $(\mathbf{X} = X_1^N, \mathbf{Y} = Y_1^N, \mathbf{Z} = Z_1^N, \mathbf{\Sigma}_= \Sigma_1^N)$ refer to *vectors* of random variables, and bold lowercase variables $(\mathbf{x} = x_1^N, \mathbf{y} = y_1^N, \mathbf{z} = z_1^N, \sigma = \sigma_1^N)$ refer to vectors of their *realizations*. The symbol "$\oplus$" stands for the *module-two addition* of binary symbols. The bold $\mathbf{H}$ stands for the parity-check *matrix* of channel code, $H(X)$ stands for the *entropy* of the memoryless source, and $H(\mathcal{X})$ [CT91] stands for the entropy-rate of the *memory* source.

More notation will be defined, when needed, for clarity of writing.

1.2 Lossless compression of correlated sources

Distributed Source coding (DSC) refers to the problem of compressing correlated sources without the need of cooperation between the source encoders, but with a joint decoder. Here, two correlated discrete sources X and Y are considered, but the results can be extended to any number, as in [LGS04, LGS06] for three correlated sources. *Prima facie*, this problem seems desperate since the well-known Shannon's Theorem [Sha59] is only proved when the two encoders cooperate to send the best joint description of the sources to the joint decoder; in that case, the minimum achievable transmission rate that allows perfect recovery is the joint entropy of the sources, $H(X,Y)$. Then we expect some rate loss when forbidding any cooperation at the encoders. However, Slepian and Wolf in 1973 [SW73], stated that no rate loss is incurred by the disjoint encoding of binary sources, and Wyner and Ziv in 1974 [Wyn74] stated a partial generalization of this result to continuous sources. In the following, we present these results and the conditions for the success of such codings.

1.2.1 The Slepian-Wolf coding paradigm

1.2.1.1 The Slepian-Wolf rate region

The Slepian-Wolf (SW) theorem [SW73], for distributed compression of two correlated discrete sources X and Y, states that the minimum achievable rate, to ensure their perfect recovery, is still $H(X,Y)$ when *no cooperation* exists between the encoders, provided the individual achievable rates R_X and R_Y verify Equation (1.1), and the decoding is done jointly.

$$\begin{cases} R_X \geq H(X|Y) \\ R_Y \geq H(Y|X) \\ R_X + R_Y \geq H(X,Y) \end{cases} \tag{1.1}$$

The minimum achievable rate for each source is its conditional entropy given the other. The equations in (1.1) can be summarized by the picture in Fig. 1.1. The so-called Slepian-Wolf rate region, in *red*, is only valid for a given correlation between the sources; the less the sources are correlated, the further it is from the origin. It can be noted that forbidding cooperation between the encoders, but allowing joint decoding, involves preventing from achieving the whole *blue* rate region in (Fig. 1.1). The difference comes from the necessity to send at least information that amounts the conditional entropy from each source, not only a total rate that amounts the joint entropy. Constructing practical codes that adapt to any correlation level between the sources is a challenging problem.

In that picture, several operating points are of our interest.

- The "corner points" (A and B in Fig. 1.1) correspond to the case where one of the sources (say Y, point A) is transmitted at a rate equal to its entropy $H(Y)$ and can be recovered error-free by the decoder without the need of any information from the other source (say X); X is transmitted at a rate equal to its conditional entropy $H(X|Y)$ and can be recovered exploiting the information from Y. This case is called the "asymmetric setup" of the SW problem (or "DSC

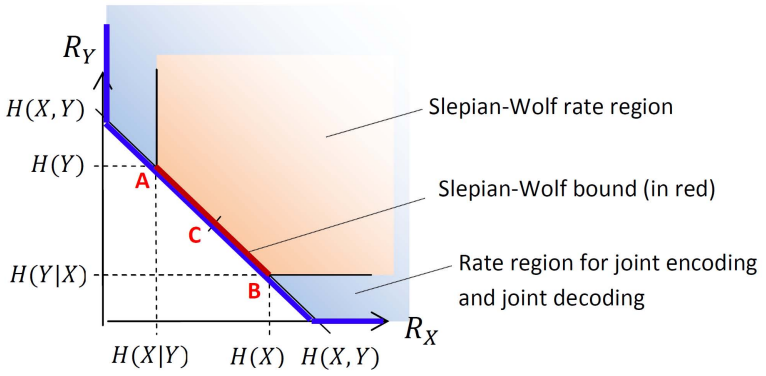

Figure 1.1: The domain of achievable compression rates for disjoint coding of correlated sources is the so-called "Slepian-Wolf rate region".

with side information at the decoder"), and Y is called the "side-information" for the decoding of X.

- The "non-asymmetric points" (between A and B in Fig. 1.1) correspond to the "non-asymmetric setup" of the SW problem. In that case, neither X nor Y is available at the decoder; the two sources are transmitted at rates R_X and R_Y verifying $H(X|Y) \leq R_X \leq H(X)$ and $H(Y|X) \leq R_Y \leq H(Y)$ respectively. The decoder must recover the two sources simultaneously.
- The "symmetric point" (point C in Fig. 1.1) corresponds to the "symmetric setup" of the SW problem; symmetric transmission rates are allocated to both sources. In this case, $R_X = \frac{1}{2}\left(H(X) - H(X|Y)\right)$ and $R_Y = \frac{1}{2}\left(H(Y) - H(Y|X)\right)$.

The *asymmetric* and the *symmetric* cases are particular cases of the *non-asymmetric* one. The asymmetric setup has found considerably more interest in terms of applicative implementations for multimedia: [LVG07b, LVG07a, VLG08] for image and audio authentication, and [PR02, AZG02, AAD+07] for Distributed Video Coding. The non-asymmetric and the symmetric setups are motivated by the application for optimal rate allocation in wireless sensor networks [RG07]. In Sections 4.1 and 4.2, we give some novel tools to improve the decoding performance of DSC codes for the asymmetric SW problem; in Section 4.3, we investigate the conditions to improve the decoding for the non-asymmetric SW problem.

1.2.2 Random code partitioning solution for DSC

To prove their landmarking Theorem [SW73], Slepian and Wolf rely on the principle of *random binning*. First, the achievability of the corner points is shown. More precisely, let N be the length of the vector of source realization that is to be compressed. Every $\mathbf{x}$ among the $2^{NH(X)}$ is randomly and uniformly assigned to one of $2^{NH(X|Y)}$ bins identified by an index. The encoder sends the index of $\mathbf{x}$. The decoder finds the estimate $\hat{\mathbf{x}}$ as the vector contained in the bin referenced by the index that is most correlated to $\mathbf{y}$. Finally, time-sharing allows to achieve all the points in the SW region. If the binning approach is shown to be optimal, it is not convenient for

practical implementation, because of exhaustive search that has to be done in lieu of the decoding. However, the proof gives some useful insights about the solution

1.2.3 Practical solutions for Asymmetric DSC

Despite its power, the SW theorem has been ignored during decades since the proof given by Slepian and Wolf in [SW73] is non-constructive. It is only lately that some solutions were proposed in the literature to perform source coding with side-information at the decoder based on channel codes. They can be split into two categories: *syndrome-based* approaches and *parity-based* approaches.

1.2.3.1 The syndrome-based approach

The optimality of this approach is proved by Wyner in [Wyn74]. It is suggested to construct bins as cosets of a capacity-achieving channel code. More precisely, an (N, K) channel code partitions the space of 2^N sequences into 2^{N-K} "cosets" containing 2^K words each, with maximum Hamming distance between them. Each coset is indexed by an $(N-K)$-long *syndrome*, which transmission to the decoder yields compression of the source. All the sequences in the same coset $\mathcal{C}_{\mathbf{s}}$ share the same syndrome $\mathbf{s}$, i.e. $\mathcal{C}_{\mathbf{s}} = \{\mathbf{x} : \mathbf{Hx} = \mathbf{s}\}$. To encode a particular vector $\mathbf{x}$, the encoder transmits its syndrome $\mathbf{s_x} = \mathbf{Hx}$, achieving a compression rate of $\frac{N-K}{N}$; the side-information $\mathbf{y}$ is sent at its entropy $H(Y)$ and can therefore be retrieved. The decoder's estimation of $\hat{\mathbf{x}}$ consists in finding the closest sequence to $\mathbf{y}$ having syndrome $\mathbf{s_x}$. If the code achieves the capacity of the underlying channel modeling the correlation between the sources X and Y, then the code should manage to retrieve $\mathbf{x}$ error-free.

The first practical asymmetric Slepian-Wolf coding solution is called DIstributed Source Coding Using Syndromes (DISCUS), and has been proposed in [PR99] using the Viterbi algorithm on a modified trellis for *systematic* Convolutional codes. Later, Roumy *et. al* [RLG07] introduced a novel representation for Convolutional and Turbo codes, based on the *syndrome trellis*, that is more suited to the syndrome approach. This approach allows to deal with *any* Convolutional/Turbo codes (not only systematic ones). The syndrome-based decoding of a Convolutional/Turbo code is presented in Section 1.3.1.

The authors of [LXG02] describe the practical implementation of syndrome-based LDPC codes; this method is still the standard decoding for such codes. The syndrome-based decoding of an LDPC code is presented in Section 1.3.2.

1.2.3.2 The parity-based approach

The parity approach offers the ability to utilize traditional channel codes without the need to implement other decoding algorithms. The main drawback of the syndrome-based approach described in Section 1.2.3.1 is its difficult implementation for puncturing the syndrome, which involves the degradation of the code [TL05a]. The parity-based approach aims at solving this issue with the use of the generator matrix of the channel code. More precisely, consider an $(N, 2N-K)$ systematic channel code, having the generator matrix $\mathbf{G} = \binom{I}{P}$ of size $(2N-K \times N)$, where I is

the identity matrix of size $N \times N$ and P is a parity-check matrix of size $(N-K) \times N$. Let $\mathbf{H}$ be the parity-check matrix s.t $\mathbf{HG} = \mathbf{0}$. To encode, the word $\mathbf{x}$ of length N is multiplied by $\mathbf{G}$, yielding the vector $[\mathbf{x}, \mathbf{x}_p] = \mathbf{Gx}$; the transmission of only the parity part $\mathbf{x}_p$ of length $(N-K)$ ensures a compression rate of $\frac{N-K}{N}$. At the decoder, the side-information $\mathbf{y}$, correlated to $\mathbf{x}$, of length N is available. The decoding is performed, using the usual ML decoder to "correct" the word $[\mathbf{y}, \mathbf{x}_p]$, knowing that $\mathbf{H}[\mathbf{x}, \mathbf{x}_p] = \mathbf{0}$, and the parity part $\mathbf{x}_p$ is perfectly known.

The principle was first introduced and implemented in [GFZ01] using Turbo codes, for the general case of non-asymmetric SW problem (Section 1.2.4). The Turbo decoding takes into account that the parity part is perfectly known in the calculation of the transition probabilities in the trellis of the code. It has then been used by [BM01, AG02, GFC04] to improve the performance of the code in terms of achieving the SW bound.

The parity approach has also been implemented for LDPC codes in [SF05]. The information that some part of the word are perfectly known is exploited by setting the corresponding LLR to infinity.

1.2.4 Practical solutions for Non-asymmetric DSC

Code design has then been recently extended to the case where both sources are compressed, in order to reach any point of the segment between A and B, see Fig. 1.1, for a given cross-over probability p. We refer to this set-up as *non asymmetric* DSC and to *symmetric* DSC, when both sources are compressed at the same rate.

1.2.4.1 Time-sharing

The approaches in Section 1.2.3 reach the asymmetric points of the SW rate region. To achieve the remaining points on the SW bound, for a given correlation level, one can implement *time sharing based* solutions. More precisely, let $\alpha \in [0, 1]$. Then, if Y plays the role of the side-information in a proportion α of some time unit, and if X plays the role of the side-information during the other $(1-\alpha)$ proportion of the same time unit, then the average transmission rates $R_X = \alpha H(X|Y) + (1-\alpha)H(X)$ and $R_Y = \alpha H(Y) + (1-\alpha)H(Y|X)$ are achieved. The resulting couple of rates (R_X, R_Y) covers all the SW bound. This solution is practically hard to implement since it requires perfect time synchronization between the two sources, otherwise the side information would be shifted, with respect to the source that has been compressed.

1.2.4.2 The syndrome-based approach

The first syndrome-based approach that is able to achieve the whole SW bound was proposed by Pradhan and Ramchandran in [PR00] using systematic Turbo codes, based on the DISCUS framework [PR99]. The authors consider an (N, K) channel code with generator matrix $\mathbf{G}$ of size $K \times N$ and partition it into two sub-codes of respective generator matrices $\mathbf{G}_1$ and $\mathbf{G}_2$ of respective sizes $k_1 \times N$ and $k_2 \times N$, s.t. $k_1 + k_2 = K$. Therefore, the total encoding rate is not modified, ensuring that the code achieves the same bound as its asymmetric version, and the correlated pairs

$(\mathbf{x}, \mathbf{y})$ can still be determined uniquely by properly choosing the rows of $\mathbf{G}$ forming the two sub-codes. Later, inspired by this work, Stankovic *et. al* extend the principle to practical punctured Turbo and IRA codes and manage to achieve the SW at a 0.04-bit gap. Systematic linear codes have also been considered in [GD05] using any linear block code, and later in [TL05b] using the so-called "syndrome formers" and "inverse syndrome formers". More precisely, the encoders send the two syndromes $\mathbf{s_x}$ and $\mathbf{s_y}$, of $\mathbf{x}$ and $\mathbf{y}$, along with k' systematic bits of $\mathbf{x}$ and $K - k'$ systematic bits of $\mathbf{y}$. The decoder first finds the difference $\hat{\mathbf{z}}$ between the two sources, and recovers the missing parts using the linearity of the code. This approach is described with more details in Section 4.3.1.1.

1.2.4.3 The parity-based approach

Existing channel codes can also be employed to cover the whole SW bound, given the correlation between the two sources. The general approach is presented by Cabarcas and Garcia-Frias in [GFC04]. $\mathbf{x}$ is partitioned into two subsets $\mathbf{x}^h = x_1^l$ and $\mathbf{x}^s = x_{l+1}^N$, where $l \in [1, N]$; the complementary partitioning is apply to $\mathbf{y}$ to obtain $\mathbf{y}^h = y_{l+1}^N$ and $\mathbf{y}^s = y_1^l$. $\mathbf{x}^h$ and $\mathbf{y}^h$ are compressed at their entropy and can be recovered at the decoder (for binary sources, these parts are simply sent to the decoder); $\mathbf{x}^s$ and $\mathbf{y}^s$ are coded by a channel encoder to yield the parity bits $\mathbf{c}^x = (c_1^x, \dots c_a^x)$ and $\mathbf{c}^y = (c_1^y, \dots c_b^x)$, where $a \geq (N - l)H(X|Y)$ and $b \geq H(Y|X)$. It can be shown that varying $l \in [1, N]$ involves reaching the whole SW bound, provided that the channel code achieves the capacity of the underlying correlation channel between the sources X and Y. The implementation of this principle has been carried out by the same authors using Turbo codes in [GFZ01, GFC04] to reach all the non-asymmetric point.

1.2.5 Practical solutions for Rate-adaptive DSC

The setup presented in the previous Sections 1.2.3 and 1.2.4 consider that the correlation between the two sources remain constant over time, which is not true in practical applications. For example for joint optimization of the rate and power of transmission in a sensor network application [RG07]. Spatially distributed sensors gather the data and send them to a common center. It is shown in [RG07] that the optimal rate allocation actually depends on the transmission conditions and can therefore be associated to any point in the SW region. It is thus of interest to construct DSC codes that can achieve any point in the SW region.

1.2.5.1 The syndrome-based approach

As the syndrome approach is barely amenable to code puncturing [TL05a], the syndrome must first be protected in order to maximize the decoding performance of each sub-code induced by the puncturing. For the asymmetric setup, it is proposed in [VAG06] to *accumulate* the syndromes of an LDPC code before the puncturing. This is equivalent to merging some rows of the parity-check matrix by adding them modulo-two. More precisely, consider an (N, K) linear block code with matrix $\mathbf{H}$ of size $(N - K) \times N$. A set of M matrices $(\mathbf{H}_m)_{m=1}^M$, with increasing sizes $(N -$

$K_m) \times N, K \leq N$, are created by merging decreasingly less rows of $\mathbf{H}$. $\mathbf{y}$ is available at the decoder and $\mathbf{x}$ is compressed by the encoder at the lowest compression rate to yield the syndrome $\mathbf{s_x}$ of size N. Knowing the correlation between the sources, the decoder tries to estimate $\hat{\mathbf{x}}$ from $\mathbf{y}$ and a small subset of $\mathbf{s_x}$, corresponding to $\mathbf{H}_1$. If the decoding fails, more syndrome bits are requested from the encoder. This process goes on until the decoding is successful. With this framework, there is no need to re-encode $\mathbf{x}$ after each failure of the decoding. This approach is extended for the whole SW bound in Section 4.3.1.

1.2.5.2 The parity-based approach

To adapt to any correlation level between the sources, standard puncturing of the parity bits can be applied. The decoding is also standard as that of channel decoding.

1.2.6 Approaches based on source codes

Besides techniques based on channel coding, a few authors have also investigated the use of *source codes* for the SW problem. This is motivated by the fact that existing source coders obviously exhibit nice compression features that should be retained, such as the ability to employ flexible and adaptive probability models, and low encoding complexity. In [KTRR03] the problem of designing a variable-length distributed source code is addressed; it is shown that the problem of designing a zero-error coder is NP-hard. In [ZE03] a similar approach is followed; the authors consider the problem of designing Huffman and Arithmetic distributed codes for multilevel sources with zero or almost-zero error probability. The idea is that, if the source and the side information are dependent, the same codeword (or the same interval for the arithmetic coding process) can be associated to multiple symbols. This approach leads to an encoder with a complex modeling stage (NP-hard for the optimal code, though suboptimal polynomial-time algorithms are provided in [ZE03]), while the decoding process resembles a classical arithmetic decoder.

In [GMO07] an extension of arithmetic coding (AC), named distributed arithmetic coding (DAC), has been proposed for asymmetric Slepian-Wolf coding. The idea is to perform the binary AC process in such a way as to allow the intervals of symbols "0" and "1" to overlap to some extent. This introduces an ambiguity in the description of the source, which lowers the codeword bit-rate, and requires a correlated side information signal to be resolved. Moreover, in [GMO09] DAC has been extended to the case of symmetric distributed coding of two sources at arbitrary rates within the Slepian-Wolf rate region. A rate-compatible extension of DAC has been presented in [GMTO08]. Similar concepts have been proposed in [AMGT07], in which the interval overlap is applied to quasi-arithmetic codes, and [AMGT08], in which sources with memory are considered.

1.3 Practical implemantation of two syndrome-based DSC codes

1.3.1 Syndrome-based Convolutional/Turbo coding

The Turbo code we use in the systems in Sections 4.1.1 and 4.3.2 is composed of two identical (n, k, L) Convolutional codes (Fig. 1.3) separated in parallel by an N-long interleaver, and the decoding is performed using the syndrome trellis first described in [RLG07]. Instead of generating parity bits as for the original Turbo code [GFZ01, GFC04], we give syndrome bits to the decoder.

1.3.1.1 The syndrome trellis

Given the constituent (n, k, L) Convolutional code of the generator matrix $\mathbf{G}$, the idea in [RG07] is to build the syndrome trellis based on the parity-check matrix $\mathbf{H}$, s.t. $\mathbf{HG} = \mathbf{0}$. This construction is of low cost in the sense that there is no need to expand the parity check polynomials matrix (see Equation (1.2)) into a matrix of an equivalent block code of large dimension. We consider here a numerical example for easier explanation. Let $\mathbf{H}$ be the parity-check matrix of the $(3, 1, 4)$ Convolutional code; the constraint length of the code is $L = 4$, $n = 3$, and $n - k = 2$. The matrix is given by:

$$\mathbf{H} = \begin{pmatrix} 11 & 15 & 06 \\ 15 & 12 & 17 \end{pmatrix}_{(oct)} = \begin{pmatrix} 1001 & 1101 & 0110 \\ 1101 & 1010 & 1111 \end{pmatrix}_{(bin)} \tag{1.2}$$

In order to efficiently construct the syndrome trellis, we derive the following diagram from the binary form of $\mathbf{H}$ given in equation (1.2). The boxes in gray represent the memories (their combined values define the *states* of the trellis).

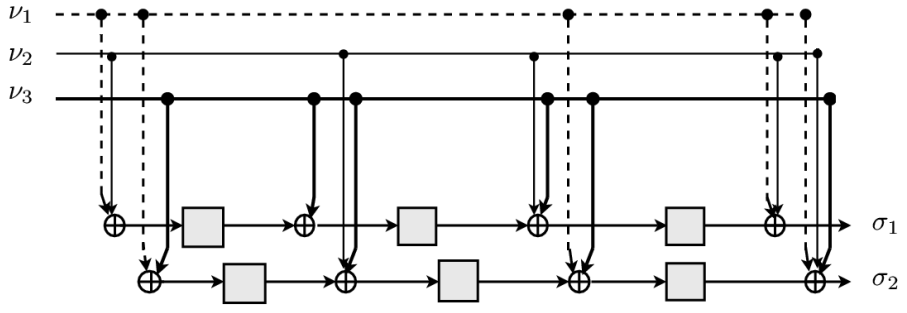

Figure 1.2: Block diagram for the computation of the syndrome for the rate $2:3$ Convolutional code.

The states of the resulting trellis are determined by the values of the memories in that diagram. The transition between two states of the trellis is labeled by three input bits (noted ν_1, ν_2, ν_3) and two output bits (noted σ_1, σ_2).

1.3.1.2 Encoding using the syndrome trellis

Let $\mathbf{x}$ be the binary word of length N that is to be compressed into its $(N-K)$-long syndrome $\mathbf{s_x}$. The coder splits $\mathbf{x}$ into small portions of length n, and feeds the block diagram in Fig. 1.2 with that information. $(n-k)$-long syndrome portions are output. This process is repeated until all the N bits are treated. This process has a linear complexity with N, and does not need to know explicitly the whole $(N-K)\times N$-sized parity-check matrix $\mathbf{H}$.

The coding can also be done by multiplying the word to be encoded with the matrix representation of the code. Therefore, $\mathbf{s_x} = \mathbf{Hx}$. However, this encoding solution has a complexity that scales with N^2.

1.3.1.3 Decoding of syndrome-based Convolutional codes

In channel coding, the standard algorithms for the decoding of convolutional codes over the BSC are the Viterbi algorithm [Vit67] and the BCJR algorithm [BCJR74], from the names of its inventors Bahl, Cocke, Jelinek, Raviv. The Viterbi algorithm performs the optimal *block-wise Maximum a posteriori* (MAP):

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}:\mathbf{Hx}=\mathbf{s_x}} \mathbb{P}(\mathbf{x}|\mathbf{y}) \tag{1.3}$$

while the BCJR algorithm performs the optimal *symbol-wise* MAP:

$$\forall n \in [1,N], \hat{x}_n = \arg \max_{x_n\in\{0,1\}} \mathbb{P}(x_n|\mathbf{y}) \tag{1.4}$$

In asymmetric DSC, when the virtual correlation channel between the correlated sources X and Y is modeled as a BSC, the Viterbi algorithm is recast into a syndrome-based algorithm that takes into account the syndrome of $\mathbf{x}$ by performing the optimal block-wise MAP

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}:\mathbf{Hx}=\mathbf{s_x}} \mathbb{P}(\mathbf{x}|\mathbf{y},\mathbf{s_x}) \tag{1.5}$$

and the BCJR is recast into a syndrome-based algorithm that performs the optimal symbol-wise MAP:

$$\forall n \in [1,N], \hat{x}_n = \arg \max_{x_n\in\{0,1\}} \mathbb{P}(x_n|\mathbf{y},\mathbf{s_x}) \tag{1.6}$$

The decoder, knowing the syndrome $\mathbf{s_x}$, looks for the sequence having that syndrome which is closest to $\mathbf{y}$ in terms of their Hamming distance.

In the work presented here, we decide to always use syndrome-based BCJR for the decoding. The recurrences for the calculation of the forward state metric, noted α in the literature, and the backward state metric, noted β, are the same as in [BCJR74]:

$$\alpha_j^t = \sum_{i\in\{\mathbf{0},\mathbf{1}\}} \alpha_i^{(t-1)} \cdot \gamma_{i,j}^{(t-1),t}$$
$$\beta_i^t = \sum_{i\in\{\mathbf{0},\mathbf{1}\}} \gamma_{i,j}^{t,(t+1)} \cdot \beta_j^{(t+1)}$$

where:

- α_j^t is the forward probability for the source to be in state j at position t;
- β_i^t is the backward probability for the source to be in state i at position t.

Basically, the only change to bring to the original BCJR decoding [BCJR74] is the calculation of the branch metric, γ.

Let $(m_t)_{t=1...\tau}$ the sequence of states of the trellis corresponding to a given block $\mathbf{x}$. Let ν_1^n be the n input bits and σ_1^{n-k} the $(n-k)$ output bits labeling the transition between the states m_{t-1} and m_t, as on Fig. 1.2. Let y_1^n be the current side information bits and s_1^{n-k} the current syndrome bits. Let p_j be the extrinsic probability $\mathbb{P}(\hat{x}_j = 1), j \in [1, N]$. By definition, for a uniform binary source, $\gamma = \mathbb{P}(m_t, \mathbf{y}_t | m_{t-1})$ is given by:

$$\gamma = \delta_{\sigma_1^{n-k}=s_1^{n-k}} \cdot \prod_{j=1}^{n} \Big(p^{\delta_{\nu_j \neq y_j}} \cdot (1-p)^{\delta_{\nu_j = y_j}} \cdot p_j^{\delta_{\nu_j=1}} \cdot (1-p_j)^{\delta_{\nu_j=0}} \Big) \tag{1.7}$$

where δ is the Kronecker's symbol ($\delta_{bool} = 1$ if $bool = true$ and 0 otherwise).

Because the transitions of the trellis that do not match the received syndrome are not followed (since $\delta_{\sigma_1^{n-k}=s_1^{n-k}} = 0$), the search is actually performed in the coset with syndrome $\mathbf{s_x}$. Note that to compress the source, the syndrome has to be punctured; that changes the term $\delta_{\sigma_1^{n-k}=s_1^{n-k}}$ of (1.7) into the following expression:

$$\prod_{i=1}^{n-k} \left(\left(\frac{1}{2} \right)^{\delta_{Pct(s_i)}} (\delta_{\sigma_i = s_i})^{\delta_{\overline{Pct(s_i)}}} \right) \tag{1.8}$$

where "$Pct(s_i)$" illustrates the information that bit i of the syndrome is punctured. Puncturing the syndrome makes our code rate compatible for other compression rates.

The first line in the product of equation (1.7) formalizes the information on the correlation from the side information; note that the process requires prior knowledge of the BSC parameter p. The second line exploits the extrinsic probabilities.

The expression of the source *a posteriori* probabilities remain the same as described in [BCJR74], $\forall j \in [1, N]$:

$$\mathbb{P}(x_j = 1 | \mathbf{y}, \mathbf{s_x}) = \sum_{i \in \{0,1\}} \alpha_i^{(t-1)} \gamma_{ij}^{(t-1,t)} \beta_j^t \tag{1.9}$$

1.3.1.4 The Turbo-syndrome framework for coding uniform Bernoulli sources

The source X, having realizations $\mathbf{x}$ of length N, is mapped into its two syndromes $\mathbf{s_{x1}}$ and $\mathbf{s_{x2}}$, of length $(N-K)$, s.t. $\frac{2 \cdot (N-K)}{N} = R_X, R_X \geq H(X|Y)$.

Then the modified BCJR is used for each constituent Convolutional decoder to estimate $\hat{\mathbf{x}}$, passing iteratively updated extrinsic messages between them, at each iteration. The Turbo decoding stops when $\hat{\mathbf{x}}$ matches the two syndromes, or when a maximum number of iterations is reached.

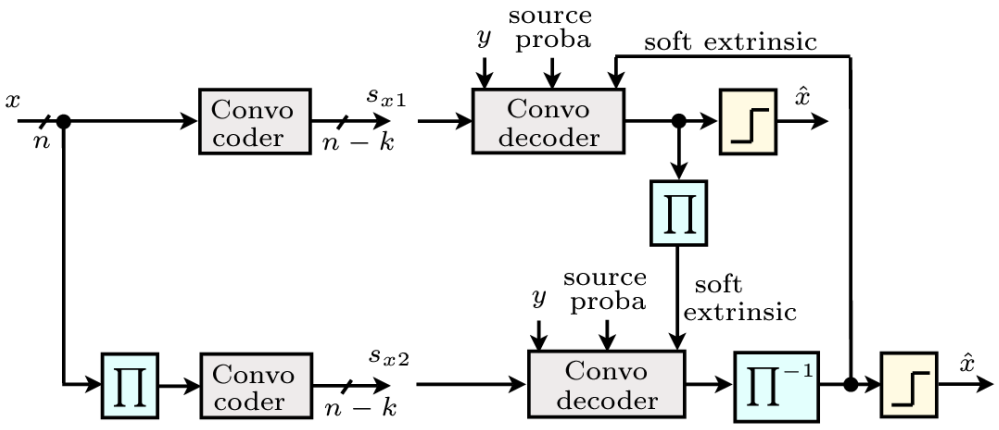

Figure 1.3: Turbo-syndrome scheme for asymmetric coding of correlated sources.

The Turbo decoder does not perform the optimal symbol-wise MAP decoding (1.4) since its graph contains cycles due to the interleaver. The decoding is sub-optimal in that sense.

1.3.2 Syndrome-based LDPC coding

The (N, K) syndrome-based LDPC codes that we use in Sections 4.1.2,4.2, 4.3.1 and 4.3.3 are defined by an $(N-K) \times N$ parity-check matrix $\mathbf{H}$ which is sparse, i.e. which contains a larger proportion of 0's than 1's. The standard syndrome-based decoding that we present here was first introduced in [LXG02].

1.3.2.1 Factor graph of a syndrome-based LDPC code

The factor graph [KFL01] of an LDPC code (or "Tanner graph" [Tan81]) graphically represents the relationships between the variables involved in the coding/decoding process. For an (N, K) code, mapping an N-long word $\mathbf{x}$ to its $(N-K)$-long syndrome $\mathbf{s_x}$, with the side-information $\mathbf{y}$ available only at the decoder, and the error pattern $\mathbf{z}$ representing the correlation between $\mathbf{x}$ and $\mathbf{y}$, Fig. 1.4 shows the corresponding factor graph.

Circle nodes represent the variables that are involved in the process. Square nodes represent the functions linking them.

1.3.2.2 Matrix representation of a syndrome-based LDPC code

The parity-check matrix of the LDPC code reflects the same relationships between the variables involved in the coding/decoding process. More precisely, the matrix $\mathbf{H} = (h_{mn})_{m \in [1,(N-K)], n \in [1,N]}$ can be deduced from the graph of the code (and reciprocally) since $\forall m, n, h_{mn} = 1$ in the matrix if, and only if, the check node s_m is connected to the variable node x_n in the graph.

The construction of the LDPC code corresponds to placing the 1's in the matrix according to a given couple of degrees distributions (see next Section).

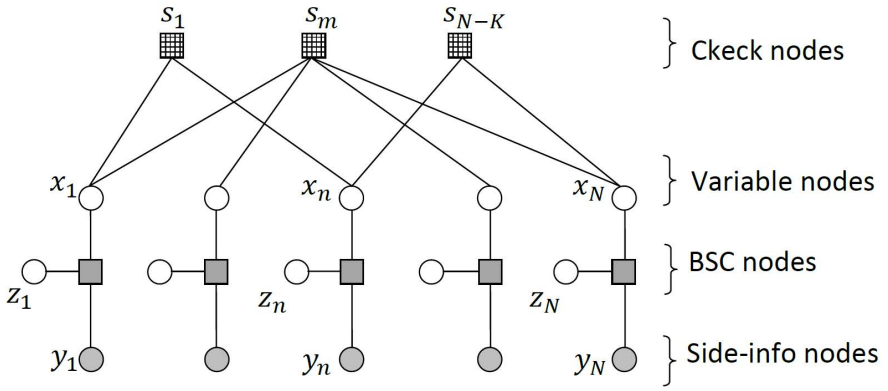

Figure 1.4: General factor graph on an LDPC code.

1.3.2.3 Degree distribution of an LDPC code

The *variable degree distribution* $\Lambda(x) = \sum_{d_v=1}^{d_v^{max}} \lambda_{d_v} x^{(d_v-1)}$ and the *check degree distribution* $\Phi(x) = \sum_{d_c=1}^{d_c^{max}} \phi_{d_c} x^{(d_c-1)}$ completely describe the LDPC code. $\forall d_v \in [1, d_v^{max}]$, there are $\lambda_{d_v} N$ columns of $\mathbf{H}$ with d_v 1's, and $\forall d_c \in [1, d_c^{max}]$, there are $\lambda_{d_c} N$ rows of $\mathbf{H}$ with d_c 1's. d_v^{max} and d_c^{max} stand for the maximum number of 1's in the columns and the rows of $\mathbf{H}$.

The optimal degree distribution of LDPC codes is function of their compression rate $\frac{N-K}{N}$. All the codes having the same length and the same degree distribution achieve the same performance with respect to the Slepian-Wolf bound. These distributions also depend on the channel that models the correlation between the sources X and Y (i.e. BSC, Gaussian, Laplacian, ...).

An open tool is available on the Internet [ldp], based on density evolution [RSU01], for finding the best degrees distributions associated to a given code rate, and theoretically informs on the gap to the SW bound that is incurred. Once the best degree distribution is found, the matrix $\mathbf{H}$ can be built using the Progressive Edge Growth (PEG) algorithm [HEA05], that we present with more details in Annex B.1.

1.3.2.4 Encoding with the syndrome-based LDPC code

As for Turbo codes, the encoding can be done by two different ways. First, the graph of the code can be used; in this case, the value of each resulting syndrome bit is the modulo-two addition of all the input bits of $\mathbf{x}$ connected to it. This encoding solution has a complexity that scales with $N \log N$. Otherwise, the encoding can be performed by multiplying $\mathbf{x}$ with the matrix $\mathbf{H}$, to yield the syndrome $\mathbf{s_x} = \left(s_m = \sum_{n=1}^{N} h_{mn} x_n\right)_{m=1}^{N-K}$; this encoding has a complexity that scales with N^2.

1.3.2.5 Decoding of the syndrome-based LDPC code

The standard decoding algorithm (also called sum-product algorithm, or belief propagation) for the syndrome-based LDPC code is proposed by Liveris et. al in [LXG02].

The original sum-product algorithm for general LDPC codes was proposed by Gallager in his Thesis [Gal62]. The aim is to find the best estimate $\hat{\mathbf{x}}$ according to the symbol-wise MAP (1.6) as for the BCJR. The decoding is sub-optimal from the presence of cycles in the factor graph of the code. To that end, extrinsic messages are propagated on the graph in Fig. 1.4. These messages are:

- d_{xn} is the degree of the variable node x_n;
- d_{sm} is the degree of the check node s_m;
- $I_n, n \in [1, N]$ are the *intrinsic*, passed from y_n to x_n;
- $E_{n,e}, n \in [1, N], e \in [1, d_{xn}]$ are the extrinsic information, passed from x_n on its e-th edge to the d_{xn} check nodes connected to it;
- $Q_{m,e}, m \in [1, (N-K)], e \in [1, d_{sm}]$ are the messages passed from s_m on its e-th edge to the d_{sm} variable nodes connected to it;
- $E_n, n \in [1, N]$ are the *a posteriori* probabilities, needed for the decision on the final estimate $\hat{x}_n$.

All these messages are Log-Likelihood Ratio (LLR), they are labeled (in) or (out) if they come *to* or *from* the nodes. For uniform binary sources, the update rules are:

$$I_n = (1 - 2y_n) \log \left(\frac{1-p}{p} \right) \tag{1.10}$$

(1.10) is the best approximation of the general formulation "$\log \left[\frac{\mathbb{P}(y_n|X_n=0)}{\mathbb{P}(y_n|X_n=1)} \right]$". Note that this calculation requires the *a priori* knowledge of the parameter p.

$$E_{n,e}^{(out)} = I_n + \sum_{k=1, k\neq e}^{d_{xn}} E_{n,k}^{(in)} \tag{1.11}$$

(1.11) is the best approximation of the general formulation "$\log \left[\frac{\mathbb{P}(X_n=0|\mathbf{y}_{\setminus n})}{\mathbb{P}(X_n=1|\mathbf{y}_{\setminus n})} \right]$".

$$Q_{m,e}^{(out)} = 2 \tanh^{-1} \left[(1 - 2s_n) \prod_{k=1, k\neq e}^{d_{sm}} \tanh \frac{Q_{m,e}^{(in)}}{2} \right] \tag{1.12}$$

$$E_n = I_n + \sum_{k=1}^{d_{xn}} E_{n,k}^{(in)} \tag{1.13}$$

(1.13) is the best approximation of the general formulation $\log \left[\frac{\mathbb{P}(X_n=0|\mathbf{y})}{\mathbb{P}(X_n=1|\mathbf{y})} \right]$.

Now there remains to decide the final estimated value:

$$\forall n \in [1, N], \hat{x}_n = \begin{cases} 0, \ if \ E_n \geq 0 \\ 1, \ if \ E_n < 0 \end{cases} \tag{1.14}$$

The decoding consists in the initialization (1.10), and repeating the operations (1.11), (1.12), (1.13), and (1.14). The decoding stops if one of the three following conditions are met:

1. $\mathbf{H}\hat{\mathbf{x}} = \mathbf{s}_{\mathbf{x}}$;

2. No symbols of $\hat{\mathbf{x}}$ are updated during the decision step (1.14);

3. The maximum number of iterations is reached (100 iterations are a good compromise between decoding performance and complexity).

1.4 Models for the correlation channel

We describe two models of binary channels that are the most commonly used models for representing memoryless channel errors (the Binary Symmetric Channel, BSC) and infinite memory channel errors (the Gilbert-Elliott, GE, channel).

1.4.1 The Binary Symmetric Channel (BSC)

1.4.1.1 Presentation of the model

The BSC of parameter $p \in [0, 0.5]$ is the simplest way to represent distortions introduced by an error-prone transmission channel. It flips the binary input $x \in \{0, 1\}$ into its complementary symbol with probability p, and it leaves the input bit unchanged with probability $(1 - p)$ (see Fig. 1.5). The capacity of this channel is given by $C_{BSC} = 1 - H(p)$, where H is the binary entropy function: $H(p) = -p\log(p) - (1-p)\log(1-p)$.

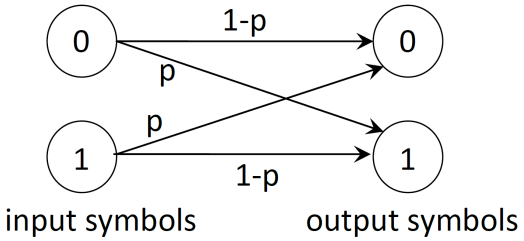

Figure 1.5: The Binary Symmetric Channel of parameter p.

Given a binary source X with entropy-rate [Cov75] $H(\mathcal{X})$, the achievable asymmetric rate for the decoding of X, given the side-information Y is

$$H(\mathcal{X}|\mathcal{Y}) = H(p) - [H(\mathcal{Y}) - H(\mathcal{X})] \tag{1.15}$$

In case X is a uniform Bernoulli source, $H(X|Y) = H(p)$.

1.4.1.2 Background on BSC parameter estimation for DSC

As described in Sections 1.3.1 and 1.3.2, the decoding of Turbo and LDPC codes require *a priori* knowledge of the BSC parameter p. In the literature, see [LXG02] or [LA05], the BSC parameter is also assumed to be available at the decoder. However, in practice, it is necessary to estimate this parameter *on-line*.

BSC parameter estimation for channel coding over the BSC In channel coding, Simons *et. al* propose an estimator of the BSC parameter in [Sim91, PSM98]. Their method consists into observing the output of the BSC, and deduce p based on the assumption that certain finite sequences appear rarely in the input. This method is only efficient when the source distribution is known.

BSC parameter estimation in the Slepian-Wolf context In the context of DSC, [GFZ01, ZRS07] propose to estimate p with an expectation-maximization (EM) algorithm. However, no pertinent initialization of the estimate is proposed, while the estimation accuracy depends on the quality of this initialization, especially for sources with low correlation (or equivalently, with large p). [FJ09] uses the Log-Likelihood Ratio, propagated during the Message-Passing of LDPC decoding, to observe a function of p; this method is only efficient for high correlation between the sources since the log-likelihoods also depend on the initial value for the estimate. Particle filtering combined with LDPC codes is used in [CWC09] to iteratively update the estimate $\hat{p}$; the method can be used to pursue slow changes of p, but it needs a large number of iterations to converge. The performance of all these methods [GFZ01, ZRS07, FJ09, CWC09] are closely dependent on the choice of initialization of $\hat{p}$, and are only efficient when the sources are highly correlated (or equivalently, with small p).

1.4.2 The Gilbert-Elliott channel

1.4.2.1 Presentation of the model

The GE is one of the simplest way to represent bursts of errors in an error-prone transmission channel. It is defined by the switch between a "good" BSC (state G) of crossover probability p_G and a "bad" BSC (state B) with crossover probability p_B, s.t. $p_G, p_B \in [0, 1]$, and $p_G \leq p_B$. The sequence of states of the channel is a Markovian process with transition probabilities g from B to G and b from G to B. The diagram for the channel generation is shown in Fig. 1.6

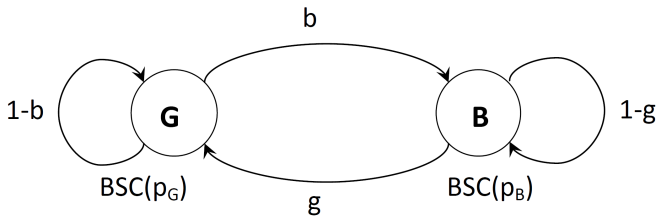

Figure 1.6: The Gilbert-Elliott channel with parameters g, b, p_G, p_B

Traditionally, the set of parameters of the model is noted $\theta = \{g, b, p_G, p_B\}$. The persistence of the channel memory can be measured using the parameter $\mu = 1-g-b$ [Ell63]; the closer $|\mu|$ is to 1, the more persistent are the states.

1.4.2.2 Achievable rates for asymmetric SW coding over the GE channel

Let Z represent the GE channel of parameter θ. Let X and Y be correlated binary sources of arbitrary distribution. The correlation between X and Y is modeled as an additive channel with $Y = X \oplus Z$. Therefore, the achievable SW bound for asymmetric coding of X given Y at the decoder is given by:

$$H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{Z}) - [H(\mathcal{Y}) - H(\mathcal{X})] \tag{1.16}$$

where the entropy-rates [Cov75] of the variables are involved. In case X is a uniform Bernoulli source, $H(X|Y) = H(\mathcal{Z})$.

A method to compute the entropy-rate $H(\mathcal{Z})$ of the GE channel has been proposed in the original paper from Gilbert [Gil60], but for practical implementation, *statistical methods* were introduced by Loeliger *et. al* to compute the information rate for finite state channels [AL01, DLV02]; it consists in generating a long random sequence $\mathbf{x}$ knowing the input distribution, and the channel output sequence $\mathbf{y}$ is obtained by simulation. The output distribution needs to be evaluated by the sum-product algorithm.

Lately, Rezaeian showed in [Rez05], based on [MBD89], that the *statistical approach* does not require the sum-product algorithm to make an estimation of the capacity of GE channels; they propose an algorithm that iteratively generates a Bernoulli random variable Z and modifies the probability of crossover q for that variable based on the outcome z (realization of Z) in each iteration l, the binary entropy of the equivalent BSC $H(q_l)$ of the random numbers generated by that *coin-tossing method* converges in probability to the entropy of the GE channel when l is large enough.

The algorithm is initialized with $q_0 = \mathbb{P}_\theta(z_0 = 1) = \frac{gp_G + bp_B}{g+b}$, which is a function of the parameters of the model.

The update rule for the crossover probability is given by:

$$q_l(\mathbf{z}_1^{l-1}) = v\left(z_{l-1}, q_{l-1}\left(z_1^{l-2}\right)\right) \tag{1.17}$$

where, assuming that $p_G < p_B$, $p_G \neq 0$, $p_B \neq 1$, and $q \in [p_G, p_B]\}$, the function $v(\cdot,\cdot)$ is given by:

$$v(z,q) = \begin{cases} p_G + b(p_B - p_G) + \mu(q - p_G)\dfrac{1-p_B}{1-q}, \ if \ z = 0 \\ p_G + b(p_B - p_G) + \mu(q - p_G)\dfrac{p_B}{q}, \ if \ z = 1 \end{cases} \tag{1.18}$$

This study of these two correlation channels ends our litterature review for lossless coding of discrete sources using SW coding. Now, we introduce lossy coding of continuous sources using WZ coding.

1.5 Rate-Distortion function for correlated continuous sources

In the context of *lossy compression*, the optimal compression is given by a *rate-distortion function*. For a given distortion between the original input and its reconstruction, the minimum compression rate is computed.

1.5.1 The Wyner-Ziv theorem for continuous sources

1.5.2 Formal expression of the rate-distortion function for general sources

First, for the compression of a single source (no side-information is needed neither at the encoder nor at the decoder), the expression of the rate-distortion function $R_X(D)$ is that described by Shannon [Sha59]; more precisely:

$$R_X(D) = \min_{p\left(\hat{X}|X\right):\mathbb{E}_{p(X,\hat{X})}\left[d\left(X,\hat{X}\right)\right]\leq D} I\left(X;\hat{X}\right) \tag{1.19}$$

where:

- D is the target distortion;
- $\hat{X}$ is the reconstructed (estimate of) X;
- $d\left(X,\hat{X}\right)$ is the measure of distortion between X and $\hat{X}$;
- $I\left(X;\hat{X}\right)$ is the mutual information between X and $\hat{X}$.

Now, consider that the side-information is available at the encoder and the decoder. Suppose that Y is this side-information and X is to be compressed. Then the rate-distortion function is given by [Ber72]:

$$R_{X|Y}(D) = \min_{p\left(X,Y,\hat{X}\right):\mathbb{E}_{p(X,Y,\hat{X})}\left[d\left(X,\hat{X}\right)\right]\leq D} I\left(X;\hat{X}|Y\right) \tag{1.20}$$

We come to the distributed source coding problem with a fidelity criterion. The side-information is only available at the decoder. Let Z be a dummy variable that is conditionally independent of X, and *s.t.* $\sum_z p(x,y,z) = Q(x,y)$. The expression of the rate-distortion function is demonstrated in [WZ76] as:

$$R^{\star}_{X|Y}(D) = \min_{p\left(Z,Y,\hat{X}\right)\in\mathbb{M}(D)} \left[I(X;Z) - I(Y;Z)\right] \tag{1.21}$$

where $\mathbb{M}(D)$ is the set of all possible distributions $p\left(x,y,z\right)$ s.t.

$$\exists f : Y \times Z \to \hat{X} : \mathbb{E}_{p(X,Y,Z)}\left[d\left(X,\hat{X}\right)\right] \leq D \tag{1.22}$$

where $\hat{X} = f(Y,Z)$.

As a corollary of the WZ theorem, Equation (1.21), it is also demonstrated in [WZ76] that:

$$R_{X|Y}(D) \leq R^{\star}_{X|Y}(D) \leq R_X(D) \tag{1.23}$$

Note that for the case of discrete sources (SW Theorem (1.1)) the equalities $R_{X|Y}(D) = R^{\star}_{X|Y}(D) = H(X|Y)$ hold and no rate loss is incurred by the disjoint encoding. In the next Section, we see under which conditions the same equalities can hold for continuous sources.

1.5.3 Zero rate loss for distributed coding of particular continuous sources

In his 1978 work [Wyn78], Wyner exhibited the conditions under which the equality $R_{X|Y}(D) = R^{\star}_{X|Y}(D)$ holds for continuous sources. These conditions are shown to hold when the sources are jointly Gaussian. In that case, a closed-form expression of the rate-distortion function is given. More precisely, suppose that $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$, $Z \sim \mathcal{N}(\mu_Z, \sigma_Z^2)$, and $Y = X + Z$; then:

$$R^{\star}_{X|Y}(D) = R_{X|Y}(D) = \begin{cases} \frac{1}{2}\log\left(\frac{\sigma_Z^2\sigma_X^2}{(\sigma_Z^2+\sigma_X^2)D}\right) & , \; if \; D \in \left]0, \frac{\sigma_Z^2\sigma_X^2}{\sigma_Z^2+\sigma_X^2}\right] \\ 0 & , \; if \; D > \frac{\sigma_Z^2\sigma_X^2}{\sigma_Z^2+\sigma_X^2} \end{cases} \tag{1.24}$$

Later in 2003, Pradhan *et. al* [PCR03] state that this equality also holds when the two correlated sources X and Y have arbitrary distributions, provided that the difference between them, Z, is Gaussian. This result comes from the information-theoretic duality between source coding with side information at the decoder, and channel coding with side information at the encoder.

To reach the rate-distortion bounds involved in the WZ problem, practical solutions consist in quantizing the continuous data, and then use channel codes to compress their binarized version. This is shown to be optimal in [KB82], in the sense that it minimizes the distortion, with respect to the other schemes.

1.6 Summary of contributions

In view of the existing work, we now present what our contributions are. This is an overview of Chapters 3 and 4 in Part II. These contributions are mostly motivated by the application to DVC (which principle is introduced in Chapter 2).

1.6.1 Source and correlation models

We consider memoryless binary sources that are *non-uniformly* distributed; the correlation is modeled as an *additive* BSC. We show that considerable gain stems from exploiting the non-uniformity, in terms of the achievable DSC rate bounds, and we propose DSC codes (based on Turbo and LDPC) that are able to estimate and to take into account the non-uniformity of the sources. In the DVC experiments, we

show that non-uniform sources are better models for the generated bit planes than uniform sources.

Similarly, we consider memory binary sources, which are modeled as Gilbert-Elliott processes [Ell63], *i.e.* with *infinite* memory; the correlation is aslo modeled as an *additive* BSC. We show that more gain comes from the exploitation of the memory, using the appropriate DSC decoding. This model appears to be a better model than uniform or non-uniform sources, when it comes to modeling the bit planes generated by DVC codecs. An EM algorithm is proposed for the joint parameter estimation and source decoding.

When the sources are modeled as non-uniform or GE sources, the conditional entropies of X and Y are no longer the same, and the *additive* correlation channel model is incomplete for the description of the correlation in the DVC setup. Therefore, we propose another correlation model, called *predictive* channel. This model explains the rate loss for the DVC experiments, since exploiting the source distribution degrades the performance of the DSC codecs when the correlation is *predictive*.

For the particular case of Binary Symmetric Channel, we propose an estimation algorithm that estimates the parameter p before the decoding. The method allows to have the same performance of the DSC codes, with respect to the distance to the SW bound that is achieved. This first estimate is then refined using an EM algorithm.

1.6.2 Tools for DSC

We propose several tools based on syndrome-based DSC codes for different purposes. First, for *asymmetric* DSC, we propose DSC codes that are able to exploit the source distributions (namely non-uniform and GE sources); these codes are able to differentiate the additive BSC from the predictive BSC. For *non-asymmetric* DSC, we propose rate-adaptive DSC codes that can reach any point of the SW bound, for a given correlation, and adapt to the correlation when it varies; this is done using accumulated and punctured DSC codes.

Particularly for non-asymmetric DSC, we raise the problem of error propagation during the decoding. This is observable for any linear code that is based on a matrix representation. We propose to limit that error propagation by appropriately designed DSC codes. Moreover, we give necessary and sufficient conditions on the codes to avoid error propagation.

Finally, for non-uniform sources, we propose a non-asymmetric DSC codec that is able to exploit the non-uniformity of the sources, while adapting to the additive and the predictive correlation channel. Necessary and sufficient conditions are also exhibited for successful recovery of the sources.

Chapter 2

Notions of Distributed Video Coding

This Chapter constitutes an overview of the existing Distributed Video Coding (DVC) techniques, and aims at introducing our contributions in this field. To get some insights about traditional video compression, we begin this Chapter with the motivation for the development of DVC tools, by presenting the state of the art of video compression before DVC. Then, we introduce what DVC brings to the domain, and we formally define the tools behind DVC. We end this Chapter by introducing our contributions to DVC.

2.1 Motivation for DVC implementation

A video sequence could be perceived as a succession of *images* (or *frames*) that give the impression of movement when put one after another. Therefore, the first video compression algorithms only compress and decompress the frames independently. However, a given frame is dependent of the previous one, which implies a large statistical time (frame to frame) and space (adjacent blocks) redundancy between the pixels. The aim of video compression is to reduce the video storage and transmission costs; this is efficiently done by exploiting that correlation. Therefore, the information contained in a particular pixel can be predicted by exploiting the information of pixels from the previous images (*inter-frame* correlation), and from adjacent pixels (*intra-frame* correlation).

MPEG-like compression algorithms perform *Discrete Cosine Transform* (DCT) on 8×8 blocks of the frames to effectively exploit *spatial correlation* within the frames (*intra* coding). This method is one among others, namely fractals, wavelets, or matching pursuit. To exploit *time correlation*, techniques based on Differential Pulse-Code Modulation (DPCM) have been proposed (*inter* coding). A clever combination of intra and inter coding techniques leads to a great data compression efficiency (*hybrid* coding).

These traditional techniques are efficient when *both* the encoder and the decoder have access to as much power as needed to perform data analysis and coding. However, emerging applications are bounded by the encoder's power restriction for the compression and large decoder's power for the decoding; for example mobile phones which send captured video sequences to a central processor must cope with the battery life duration, and hence need to perform as less operations as possible

to improve their power autonomy; such power restriction can be encountered when dealing with distributed sensor networks.

In order to deal with that low power constraint at the encoder, video compression has found interests in DSC, to yield the field of DVC. The main concern in DVC is to reduce the complexity of the encoding, which implies to perform only the most basic operations at the encoder side. This means in particular that the exploitation of the correlation is performed at the decoder only. Practical DVC solutions are based on channel codes, which improves the codec's resistance to transmission errors, or prevents error propagation between the frames.

2.2 Distributed video coding solutions

The first practical DVC solution has been proposed by Puri and Ramchandran [PR02] at Berkeley University, in 2002, under the name *Power-efficient, Robust, hIghcompression, Syndrome-based Multimedia coding* (PRISM). In the meantime, another DVC solution has emerged from the Stanford University [AZG02], designed by Girod *et. al.* In the sequel, we focus on a third DVC solution, developed within the European project *Distributed Coding for Video Services* (DISCOVER) [AAD+07, dis], which is based on the Stanford framework, and will be referred to as the *DISCOVER codec* in the sequel. All our contributions (namely in Chapter 5) aim at improving the rate-distortion performance of this last video codec.

2.2.1 Overview of the DISCOVER codec

This Section briefly describes the encoder and the decoder constituents of the DISCOVER codec. These constituents are detailed in the next two Sections 2.2.2 and 2.2.3. Fig. 2.1 shows the DISCOVER codec block diagram.

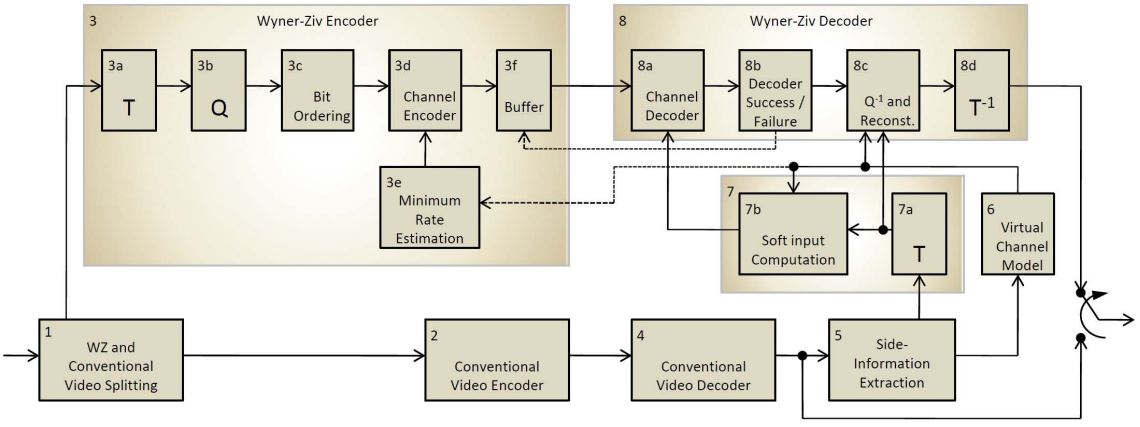

Figure 2.1: Block diagram of the DISCOVER codec.

The *encoder* first splits the video frames into key frames and WZ frames. The key frames are conventionally encoded using H264/AVC encoder and transmitted to the decoder. The WZ frames are first transformed with a Discrete Cosine Transform

(DCT), and the obtained transform coefficients are quantized. The quantized coefficients are organized into bands, where every band contains the coefficients associated to the same frequency in different blocks. Then, the quantized coefficients bands are fed bit plane by bit plane to a SW encoder, which computes their syndromes.

At the *decoder*, the key frames are first decoded using a conventional video decoder. Then a motion compensated interpolation between every two closest key frames is performed, in order to produce the SI for intermediary WZ frames. Each WZ frame is split into blocks, which are then DCT transformed. The correlation channel between the WZ and SI DCT coefficients is approximated by a Laplacian model. Knowing the side-information and the syndrome bits, the decoder estimates the transmitted bit planes. If the number of syndrome bits is not sufficient to have a BER lower than 10^{-4} at the output of the decoder, more syndrome bits are requested from the encoder.

2.2.2 The encoding process

2.2.2.1 WZ/conventional video splitting

The successive images of the video sequence are split into Group Of Pictures (GOP) including *one* key frame and several WZ frames (Block 1 in the diagram of Fig. 2.1). The GOP size can be of fixed or variable length, according to the configuration desired by the user. In case the GOP size is variable, the algorithm proposed by Ascenso *et. al* [ABP06] is run to dynamically decide the appropriate length. The algorithm proposed in [ABP06] proceeds in two steps: first, activity measures are computed between two consecutive frames, and based on that information, GOP are formed, with a constraint on the maximum GOP size. Once labeled, the key frames are intra-encoded using a traditional video encoder (MPEG,H26x/AVC, Block 2), which exploits only the spatial correlation and does not perform any motion estimation nor memory modeling. The WZ frames are WZ-coded (Block 3) to, eventually, yield syndrome bits from the SW (Low-Density Parity-Check) LDPC encoder.

2.2.2.2 DCT transform, quantization, and binarization

Each WZ frame of the GOP undergoes DCT transforms on each block of pixels of size 4×4 (Block $3a$), to yield 16 frequency bands grouping the coefficients of the same frequency. The frequency bands are then quantized (Block $3b$). The quantization level is the same for all the images of the video sequence, and depends on the target quality for the WZ frames. Next, the quantized DCT coefficients are ordered bit plane by bit plane (Block $3c$) and fed to the LDPC encoder.

2.2.2.3 SW encoding

The SW coding is rate-adaptive with respect to the correlation. The DISCOVER codec has a collection of LDPC matrices that form a rate-adaptive SW codec; they are obtained by gradually expanding, see [VAG06], an LDPC matrix of minimum rate until a square matrix of rate 1 is obtained (in that configuration, one syndrome bit is transmitted to the decoder for one source bit, Block $3d$). A full-rate syndrome

is computed for each bit plane, and placed into a buffer (Block $3f$). The minimum number of syndrome bits is then transmitted, and the decoding is performed with that information, knowing the side-information. The theoretical minimum number of syndrome bits is estimated from the side-information itself (Block $3e$), depending on the target reconstruction quality of the WZ frames.

2.2.3 The decoding process

2.2.3.1 Side-information extraction

First, the key frames are conventionally decoded (Block 4). Then, the side-information frame for the current WZ frame is extracted (Block 5) from the two most adjacent frames (*reference frames*, X_B and X_F. If the GOP is of size 2, the reference frames are the key frames directly before and after the WZ frame. If the GOP is larger, the reference frames are chosen as indicated in [ABP06]. More precisely, a motion estimation algorithm is run between the low-pass-filtered versions of X_B and X_F, so as to minimize a criterion that is expressed in [ABP06]; the low-pass filter is to reduce the perturbation from the noise that could deteriorate the resulting *motion vectors*, even for a relatively large search window (± 32 pixels). The obtained bidirectional motion vectors between X_B and X_F are exploited to generate the side-information (SI) frame Y for the WZ frame, that is the best estimate of the WZ frame so far. Each pixel $\mathbf{s}$ of the SI is computed as follows:

$$Y(\mathbf{s}) = \frac{t_F X_B(\mathbf{s} + \mathbf{u}_B) + t_B X_F(\mathbf{s} + \mathbf{u}_F)}{t_F + t_B} \tag{2.1}$$

where $(\mathbf{u}_B, \mathbf{u}_F)$ is the vector associated to the block containing pixel $\mathbf{s}$; and t_B, t_F are the distances between the WZ frame and the frames X_B, X_F respectively, when the GOP size is 2, $t_F = t_B = 1$.

2.2.3.2 Correlation channel modeling

The side-information image undergoes the same process as the WZ frame (Section 2.2.2.2) to yield the associated DCT transform coefficients (Block $7a$). In the DISCOVER codec, the difference between the WZ coefficients and the SI's coefficients is modeled as a *Laplacian* channel (such a modeling is motivated by the analysis shown in [AZG02]). However, as the WZ frame is not available at the decoder, the Laplacian parameters needed for the decoding of the WZ frame are estimated *on-line* directly from the residual R (2.2) (Block 6). In the following, we describe a precise modeling of the Laplacian channel at the coefficient level.

First, the pixel $\mathbf{s}$ of the "residual" frame R is calculated, given the motion vector $\mathbf{u} = (\mathbf{u}_B, \mathbf{u}_F)$, as:

$$R(\mathbf{s}) = \frac{X_B(\mathbf{s} + \mathbf{u}_B) - X_F(\mathbf{s} + \mathbf{u}_F)}{2} \tag{2.2}$$

Once the 16 frequency bands $R_k, k \in [1, 16]$ are formed from R, let $r_{k,i}$ be the i-th coefficient of the band k, of size N_k. Let $\mu_{|k|} = \sum_i |r_{k,i}|$. The variance $\sigma^2_{|k|}$ of the absolute values of the coefficients is thus given by:

$$\sigma^2_{|k|} = \mathbb{E}\left[|r_{k,i} - \mu_{|k|}|\right]^2 = \frac{1}{N_k}\sum_{i}^{N_k}\left[|r_{k,i} - \mu_{|k|}|\right]^2 \tag{2.3}$$

Therefore, for the band k, the parameter α_k of the Laplacian model is given, depending on the obtained value of $|r_{k,i}|$, by:

$$\hat{\alpha}_{k,i} = \begin{cases} \hat{\alpha}_k = \sqrt{\dfrac{2}{\sigma_k^2}}, & if \ \left[|r_{k,i} - \mu_{|k|}|\right]^2 \leq \sigma^2_{|k|} \\ \sqrt{\dfrac{2}{\left[|r_{k,i} - \mu_{|k|}|\right]^2}}, & if \ \left[|r_{k,i} - \mu_{|k|}|\right]^2 > \sigma^2_{|k|} \end{cases} \tag{2.4}$$

where σ_k^2 is the variance of the coefficients $r_{k,i}$.

An alternative to the Laplacian correlation model is introduced in [BKW08] to capture occlusions and objects introduction in the scene (Gaussian-Bernoulli-Gaussian channel), or when the camera is moving (Gaussian-Erasure channel). The authors exhibit the bounds on the rate-distortion functions for both models.

2.2.3.3 Rate-adaptive SW decoding

The LDPC decoding that is performed here is the Belief Propagation described in Section 1.3.2.5, but the correlation channel is no longer a BSC, but results from the combination of *Laplacian channel*, the *quantization* and the *binarization* processes (see Section 2.2.2.2). Given the subset of syndrome bits received from the syndrome buffer (Block $3f$), the LDPC decoder tries to recover the current bit plan (Block $8a$). If the decoding fails (this decision is taken in Block $8b$), the decoder requests for more syndrome bits and tries again. This goes on until the decoded bit plane is well estimated (the criteria for such good decoding are enumerated in Section 1.3.2.5). The less the frames are correlated, the more syndrome bits are requested. Note that, when all the syndrome bits are transmitted, the WZ and the SI bands are not correlated.

The search for the minimal rate to begin the rate adaptive decoding is crucial for the SW codec, since sending the minimum syndrome bits possible incurs a large number of requests, which increases the decoder's complexity. A first approach is described has been proposed in [Laj06] that is based on the WZ bound of each bit plane, taking into account the distortion in the reconstruction. Later, in [Kub08], a better lower bound of the minimum rate is computed based on the SW bound. This latter method is better since it provides a lower rate bound that is closer to the final rate of the rate-adaptive decoding. In the following, we briefly describe the method in [Kub08].

The aim is to estimate an average crossover probability $\hat{p}_{cr}$ for the whole bit plane b. The indicator function of the crossover event is given by:

$$I_{cr} = \begin{cases} 1, \ if \ x^b_{k,i} \neq \arg\max\limits_{t\in\{0,1\}} \mathbb{E}_{p_{y|x_{k,i}}}\left[\mathbb{P}\left(x^b_{k,i} = t | y^b_{k,i}, x^1_{k,i}, x^{b-1}_{k,i}\right)\right] \\ 0, \ otherwise \end{cases} \tag{2.5}$$

where $x^b_{k,i}$ is the i-th bit of the bit plane b of the coefficient band k, and $y^b_{k,i}$ is the corresponding SI.

It is shown that the average crossover probability of the bit plane is given by:

$$\hat{p}_{cr} = \frac{1}{N} \sum_i I_{cr}(x_{k,i}) \tag{2.6}$$

where N is the size of the bit plane.

The minimal theoretical rate is then estimated as:

$$R_{min} = -p_{cr} \log(p_{cr}) - (1 - p_{cr}) \log(1 - p_{cr}) \tag{2.7}$$

that is the entropy $H(p_{cr})$ of a virtual BSC which crossover probability is equivalent to that implied by the correlation channel.

2.2.3.4 Inverse quantization, reconstruction, and multiplexing

Once all the decoded bit planes of the current frame are available, inverse quantization is performed to yield estimates of the quantized DCT coefficients. A reconstruction algorithm is then run on these quantized versions to yield estimates of the original coefficients (Block 8c); in the DISCOVER codec, this reconstruction is performed according to the optimal algorithm for the Laplacian model. More precisely, let $x_{k,i}$ be the current coefficient to be decoded and $y_{k,i}$ the side-information available at the decoder, let $[l, u)$ be the quantization interval for $x_{k,i}$, let $\gamma = y_{k,i} - l$, $\delta = u - y_{k,i}$, and $\Delta = u - l$. Then the optimal estimator for $x_{k,i}$ is given by:

$$\hat{x}_{k,i} = \begin{cases} l + \dfrac{1}{\hat{\alpha}_{k,i}} + \dfrac{\Delta}{1 - e^{\hat{\alpha}_{k,i}\Delta}}, & if\ y_{k,i} < l \\ y_{k,i} + \dfrac{\left(\gamma + \frac{1}{\hat{\alpha}_{k,i}}\right) e^{-\hat{\alpha}_{k,i}\gamma} - \left(\delta + \frac{1}{\hat{\alpha}_{k,i}}\right) e^{-\hat{\alpha}_{k,i}\delta}}{2 - \left(e^{-\hat{\alpha}_{k,i}\gamma} + e^{-\hat{\alpha}_{k,i}\delta}\right)}, & if\ y_{k,i} \in [l, u) \\ u - \dfrac{1}{\hat{\alpha}_{k,i} - \frac{\Delta}{1 - e^{\hat{\alpha}_{k,i}\Delta}}}, & if\ y_{k,i} \geq u \end{cases} \tag{2.8}$$

The proof for the optimality of this reconstruction (2.8) is given in [KNG07]. The WZ decoding ends with applying inverse DCT transform to the obtained coefficients to yield the reconstructed WZ frames. Finally, appropriately multiplexing the key and the WZ frames yields the complete video sequence.

2.3 Summary of contributions

Our work on DVC, presented in Chapter 5, aims at improving the rate-distortion performance of the existing DVC codec DISCOVER. To that end, we study the binary distribution of the bit planes that are generated during the encoding step, before the syndromes are computed by the SW encoder. More precisely, we show that these bit planes can be viewed as realizations of Bernoulli or Hidden Markov sources, and we modify the SW decoder in order to take into account their distribution.

2.3.1 Non-uniform source modeling

In the first part of Chapter 5 (Section 5.1), we show that the bit planes can be modeled as realizations of non-uniform Bernoulli sources, which parameters differ from bit plane to bit plane, and need to be estimated on-line. This first model is presented in more details in Section 3.1. Then, we use the tools based on LDPC codes, presented in Section 4.1.2, to implement them in the existing DISCOVER codec. Finally, we show that considerable gain is observed when modeling the video bit planes that way. However, there is an issue, since almost all the bit planes follow some non-uniform Bernoulli distribution, but no rate gain is observed for many of them. Such a phenomenon is explained in Sections 3.3 with the introduction of the *predictive* correlation channel, and 5.1.2.4 with its analysis for the coding of video bit planes.

2.3.2 Hidden Markov source modeling

The second part of Chapter 5 is dedicated to the study of modeling the bit planes as independent realizations of hidden Markov sources (Section 5.2). We analyze the burst length distributions of the bit planes, and we show that those burst lengths are more likely observed from GE sources than Bernoulli sources, or uniform sources. Strengthened by that result, we use the tools presented in Section 4.2 for the SW coding of GE sources, to implement them in lieu of the standard channel decoding performed by the original DISCOVER. We show that the *predictive* correlation channel also applies for the GE model. We show that this modeling is more effective than non-uniform sources, or "simple" Markov sources.

For both the non-uniform and the hidden Markov source modeling, the extra decoding cost involved by the parameter estimation remains negligible, with respect to the improvement in terms of rate-distortion performance. The improvement is up to 5.7% for the non-uniform source modeling, and it is up to 10.4% for the hidden Markov modeling.

Part II

Contributions

Chapter 3

Source and correlation models for DSC

In this chapter, we first propose two source models (In Sections 5.1.1 and 5.2.1, we will show that these models are well suited for the bit planes generated by the DISCOVER codec), namely *non-uniform* Bernoulli source in Section 3.1, and *Gilbert-Elliott* (GE) [Gil60, Ell63] source in Section 3.2. We formally and empirically show the compression rate gain that can be achieved for these models, compared to the compression rate achieved for uniform sources. Then, we investigate two models for the Binary Symmetric Channel (BSC), in Section 3.3, that are used to model the correlation between correlated binary sources, namely *additive* and *predictive* BSC. We show that the expected gain from the two source models is only true when the BSC is *additive*, and some loss is always observed if a mismatch between these two models occurs during the decoding. We also present a simple and optimal method for the BSC parameter estimation, when the sources are coded with a syndrome-based DSC code. The estimation is independent of the source distribution and can be performed prior to decoding (Section 3.4). The method is based on the probability of observing *ones* in the syndromes of the sources.

3.1 Non-uniform source modeling

All DVC codec implementations often assume the binary distribution of the bit planes to be uniform, which is actually far from being the case (as we will see in Section 5.1). In this Section, we introduce the theory behind the non-uniform modeling of the binary sources and show that, interestingly, the achievable compression rates are lower than that of uniform sources. This implies that a given DSC code can successfully decode correlated non-uniform sources when the correlation is lower (with respect to the case when the source is uniform). In Sections 4.1.1 and 4.1.2, we describe the modifications to add to existing DSC codes to take into account the non-uniformity of the source.

3.1.1 The non-uniform source model in the asymmetric DSC

Let the notation "$X \sim \mathcal{B}(p_X)$" denote a binary variable, Bernoulli distributed, with parameter $p_X = \mathbb{P}(X = 1)$. In asymmetric DSC, a second source Y, with realization $\mathbf{y}$, is correlated to X and available at the decoder. The correlation is modeled as a virtual (X, Y, p) additive BSC as defined below. The BSC is the most commonly used correlation channel when dealing with correlated binary sources. The input symbols are flipped with "*cross-over*" probability p.

Definition 3.1. *An (X, Y, p) additive BSC is a channel with binary input X, binary output Y. The noise $Z \sim \mathcal{B}(p)$ is independent of the channel input, and the channel output is obtained by $Y = X \oplus Z$.*

In DSC, the achievable lower bound depends not only on the correlation between X and Y, but also on their respective distributions. Let $X \sim \mathcal{B}(p_X)$ and $Y \sim \mathcal{B}(p_Y)$. Let Z be the noise, modeled as a BSC. First, when the source X is uniform, i.e. $p_X = 0.5$, the source Y is also uniform, and

$$\begin{aligned} &H(X) = H(Y) = 1 \\ &H(X|Y) = H(Y|X) = H(Z) = H(p) \end{aligned} \tag{3.1}$$

Now, consider that X is non-uniform (i.e. $p_X \neq 0.5$). Then Y is also non-uniform. We claim the following result to characterize the rate gain that is expected when the BSC is *additive*.

Claim 3.1. *Let $X \sim \mathcal{B}(p_X)$ and $Y \sim \mathcal{B}(p_Y)$ be two correlated sources, where the correlation is modeled as a virtual (X, Y, p) additive BSC. We consider the asymmetric distributed problem, where Y is available at the decoder and compressed at its entropy $H(Y)$ and X is compressed at its conditional entropy $H(X|Y)$. If the source X is non-uniform, the compression rate for X is reduced by $H(Y) - H(X) \geq 0$ compared to the compression rate achieved for uniform sources.*

Proof. Since the BSC is additive, $\exists\ Z$ independent of X s.t. $X \sim \mathcal{B}(p_X)$ and $Y = X \oplus Z$. Then the non-uniformity of Y is characterized by the parameter $p_Y = p_X(1-p) + (1-p_X)p$, with p_Y nearer to 0.5 than p_X. The concavity of $H(\cdot)$ implies $H(Y) \geq H(X)$. Moreover $H(X|Y) = H(Z) - [H(Y) - H(X)]$ since Z is independent of X. Since $H(Y) - H(X) \geq 0$ (with equality *iff* the source X is uniform), the non-uniformity of X reduces the lower bound $H(X|Y)$, and the rate gain is $H(Y) - H(X)$. □

3.1.2 Compression rate gain

We consider *three* Bernoulli sources of respective parameters $p_X = \{0.5, 0.2275, 0.15\}$, and we compare in Fig. 3.1 the minimum achievable rates $H(X|Y)$ for each source, for the BSC parameter $p \in]0, 1[$.

It is clearly shown that the minimum achievable rates decrease when the source parameter is further from 0.5, i.e. the non-uniformity increases, for any value of

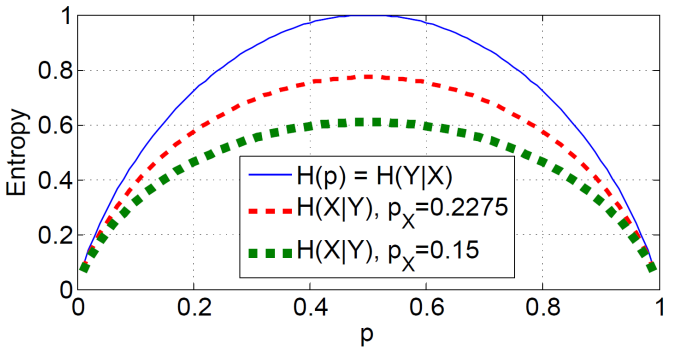

Figure 3.1: Minimum achievable rates for distributed coding of *three* non-uniform sources, for $p \in [0, 1]$.

p. Therefore, one can achieve lower compression rates when coding non-uniform sources, with respect to the compression rates achievable for uniform sources.

3.1.3 Non-uniform source parameter estimation

Let $X \sim \mathcal{B}(p_X)$, and let $\mathbf{x} = x_1^N$ be its realization of length N. We want to estimate p_X from the observation of $\mathbf{x}$ only. From [Kay93, Theorem 5.1], $\sum_{n=1}^{N} x_n$ is a sufficient statistic for the estimation of p_X., and the Minimum Variance Unbiased (MVU) estimator of p_X, with respect to $\mathbf{x}$, is:

$$\hat{p}_X = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{3.2}$$

3.2 Gilbert-Elliott source modeling

Now we investigate memory sources where the memory is spread over the entire sampled data. More precisely, the sequence of symbols is generated by a Hidden Markov Model (HMM): the two-state Gilbert-Elliott (GE) process [Ell63]. The probability of a given symbol is dependent only on the current state. The GE channel was investigated by Garcia-Frias [GF04] as a model for the correlation between X and Y and the decoding of Low-Density Parity-Check (LDPC) codes over the GE channel is also shown; here, we use it as a model for the source. We show here that a considerable gain results by exploiting the memory in the data and this gain is even greater than exploiting only the non-uniformity of the source (as in Section 3.1). The estimation of the GE channel parameters was carried out by [MBD89]. We propose the estimation of the model parameters using an *Expectation Maximization* (EM) algorithm [Rab89]. This memory source model is motivated by the DVC application; we will show that the proposed source model fits to the generated video bit planes (Section 5.2), and we will propose a modified LDPC-based EM algorithm to decode the memory source (Section 4.2).

3.2.1 The hidden Markov source model in the asymmetric DSC

Let Σ be a finite Markovian process with memory order one, having two realizations (called *states*) $\mathbf{0}$ and $\mathbf{1}$. Let X be a binary source which is dependent on the underlying and persistent Markov state process Σ. In each state $\mathbf{0}$ and $\mathbf{1}$, the source is drawn according to a Bernoulli law of parameter $p_{\mathbf{0}}$ and $p_{\mathbf{1}}$ respectively (Fig 3.2).

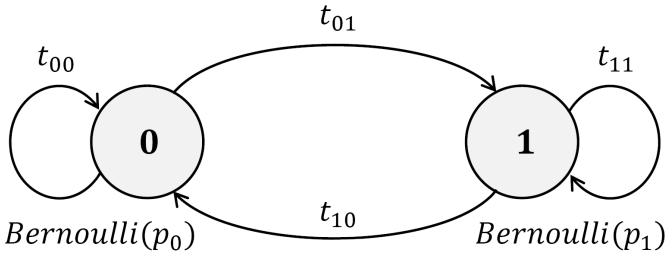

Figure 3.2: Diagram for the GE source modeling.

We define the transition probabilities $t_{\mathbf{00}}$, $t_{\mathbf{01}}$, $t_{\mathbf{10}}$ and $t_{\mathbf{11}}$, between the states, as shown in Fig. 3.2. Since $t_{\mathbf{00}} = (1 - t_{\mathbf{01}})$ and $t_{\mathbf{11}} = (1 - t_{\mathbf{10}})$, the set of parameters of the model is $\theta_X = (p_{\mathbf{0}}, p_{\mathbf{1}}, t_{\mathbf{10}}, t_{\mathbf{01}})$. These parameters are formally defined by:

$$\begin{aligned} p_{\mathbf{0}} &= \mathbb{P}_{\theta_X}(X_n = 1 | \Sigma_n = \mathbf{0}) \\ p_{\mathbf{1}} &= \mathbb{P}_{\theta_X}(X_n = 1 | \Sigma_n = \mathbf{1}) \\ t_{\mathbf{10}} &= \mathbb{P}_{\theta_X}(\Sigma_n = \mathbf{0} | \Sigma_{n-1} = \mathbf{1}) \\ t_{\mathbf{01}} &= \mathbb{P}_{\theta_X}(\Sigma_n = \mathbf{1} | \Sigma_{n-1} = \mathbf{0}) \end{aligned} \tag{3.3}$$

where $\mathbf{\Sigma} = \Sigma_1^N$ is an N-long state sequence, and $\sigma = \sigma_1^N = \{\mathbf{0}, \mathbf{1}\}^N$ is its realization.

These equations lead to the following property of the source:

Property 3.1. *$\forall n \in [1, N]$, given the state Σ_n, X_n is a memoryless Bernoulli process of parameter $p_{X_n} = \mathbb{P}_{\theta_X}(X_n = 1 | \sigma_n)$, where $p_{X_n} = p_{\mathbf{0}}$ if $\sigma_n = \mathbf{0}$, and $p_{X_n} = p_{\mathbf{1}}$ if $\sigma_n = \mathbf{1}$.*

In DSC, a second source Y with realization $\mathbf{y}$ is correlated to X, where the correlation is modeled as a virtual BSC with parameter p. We assume that $Y = X \oplus Z$, with $\mathbb{P}(Y \neq X) = \mathbb{P}(Z = 1) = p$. Given the characteristics of the model, we state the following Lemma 3.1 which characterizes the GE nature of the side-information Y.

Lemma 3.1. *Let X be a binary source drawn by a GE process of parameter $\theta_X = (p_{\mathbf{0}}, p_{\mathbf{1}}, t_{\mathbf{10}}, t_{\mathbf{01}})$. Σ denotes the underlying hidden process generating X. Let Y be a source correlated to X according to the additive channel model $Y = X \oplus Z$, where Z is a binary process.*
If the correlation noise Z is a (memoryless) Bernoulli process of parameter p, then Y is a GE source with the same underlying state process Σ.

Proof. When conditioned on the state process Σ, the source X is memoryless according to Property 3.1. Since Z is memoryless too, the source Y is memoryless when conditioned on the same state process Σ. More precisely, $\forall n \in [1, N]$, given the state Σ_n, Y_n is a memoryless Bernoulli process of parameter $p_{Y_n} = p_{X_n}(1-p)+(1-p_{X_n})p$, where p_{Xn} is described in Property 3.1. Therefore Y is a Gilbert Elliott source with the same state process as for X. □

We now turn back to the asymmetric DSC problem, where Y is available at the decoder only. As the sources have infinite memory, X can be compressed at its conditional *entropy-rate* [Cov75] $H(\mathcal{X}|\mathcal{Y}) = \lim_{N\to\infty} \frac{1}{N} H(\mathbf{X}|\mathbf{Y})$. Here, $H(\mathcal{X}|\mathcal{Y}) = H(Z) - [H(\mathcal{Y}) - H(\mathcal{X})]$. The entropy-rates $H(\mathcal{X})$ and $H(\mathcal{Y})$ of the correlated GE sources can be efficiently estimated with the statistical-based algorithm described in [Rez05].

3.2.2 Entropy estimation for a Gilbert-Elliott source

The estimation of the entropy of a GE source X is similar to the problem of estimating the capacity of the equivalent GE channel Z, having the same parameters, which has been further investigated in the literature. This has been presented in Section 1.4.2.2.

These algorithms only require the knowledge of the parameter θ. As the algorithm in [Rez05] yields good results and remains of low complexity, we have used it to estimate the entropy-rates of the GE sources: $H(\mathcal{X})$ and $H(\mathcal{Y})$.

3.2.3 Compression rate gain

Let the notation "$X \sim GE(\theta_X)$" denote a binary variable which is drawn according to the GE process, with parameter θ_X. The correlation between X and Y is modeled as a virtual (X, Y, p) *additive* BSC as defined is Section 3.1.1. We characterize in Lemma 3.2 the rate gain that is expected for the compression of the GE source X in the DSC setup, with respect to the compression rate of a uniform source.

Lemma 3.2. *Let X and Y be two correlated sources, where the correlation is modeled as a virtual (X, Y, p) additive BSC.*
If the source X is not uniform, the minimum coding rate for X is $H(\mathcal{X}|\mathcal{Y}) = H(Z)-[H(\mathcal{Y}) - H(\mathcal{X})]$. Since $H(\mathcal{Y})-H(\mathcal{X}) \geq 0$, $H(\mathcal{X}|\mathcal{Y})$ is reduced by $H(\mathcal{Y}) - H(\mathcal{X})$ compared to the minimum coding rate $H(\mathcal{X}|\mathcal{Y}) = H(Z)$ for a uniform source.

Proof. Since the BSC is additive, $\exists$ Z independent of X s.t. $Y = X \oplus Z$. If X is uniform, so is Y, and $H(\mathcal{X}|\mathcal{Y}) = H(Z)$.

Now, consider the case of an arbitrary process X. On the one hand we have $H(\mathcal{X}) = H(\mathcal{X} \oplus Z|Z)$, and on the other hand we have $H(\mathcal{Y}) = H(\mathcal{X} \oplus Z)$; this implies $H(\mathcal{X}) \leq H(\mathcal{Y})$. The conditional entropy-rate of the source X is computed as $H(\mathcal{X}|\mathcal{Y}) = H(Z) - [H(\mathcal{Y}) - H(\mathcal{X})]$, since the noise Z is independent of X. As $H(\mathcal{Y}) - H(\mathcal{X}) \geq 0$ (with equality *if, and only if,* the source X is uniform), the lower

transmission rate bound $H(\mathcal{X}|\mathcal{Y})$ is lower than that of a uniform source, and the rate gain is given by $H(\mathcal{Y}) - H(\mathcal{X})$. □

Fig. 3.3 shows the theoretical rates $H(\mathcal{X}|\mathcal{Y})$ that can be achieved for the parameter $\theta_X = (t_{\mathbf{10}} = 0.03, t_{\mathbf{01}} = 0.01, p_{\mathbf{0}} = 0.07, p_{\mathbf{1}} = 0.7)$, when p varies in $]0, 1[$, and the BSC is *additive*. The achievable coding rate decreases when X is GE source, with respect to the rate of a uniform source. Therefore, knowing the GE source distribution, one can achieve higher compression rates using the same DSC code.

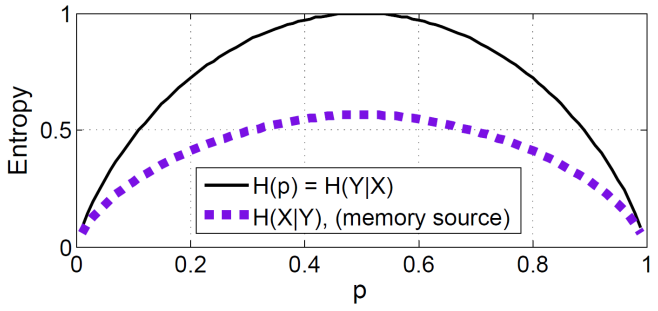

Figure 3.3: Comparison of the source conditional entropies, when X is uniform and when X is a GE process.

The LDPC-based decoder that we present in Section 4.2 is conceived to reach the theoretical bound in Fig. 3.3, by iterative update of the source parameters and the states values.

3.2.4 Parameter estimation for GE sources: The Baum-Welch algorithm

The *Baum-Welch algorithm* is an iterative maximization algorithm that aims at estimating the parameters of a Hidden Markov process, when the realization symbols of the process are observed. Here, we use it for the parameter estimation of the Gilbert-Elliott process. The aim is to find the best estimates $\hat{\theta}_X$ and $\hat{\sigma}_{\mathbf{x}}$ of θ_X and $\mathbf{\Sigma}_{\mathbf{x}}$, given the N-long realization vector $\mathbf{x}$. This is a particular case of the *Expectation-Maximization* (EM) algorithm [Rab89]. The maximization problem can be formulated as:

$$(\hat{\sigma}_{\mathbf{x}}, \hat{\theta}_X = \arg \max_{(\sigma_{\mathbf{x}}, \theta_X)} \mathbb{P}_{\theta_X}(\mathbf{x}, \sigma_{\mathbf{x}}) \tag{3.4}$$

No expression of the solution to that problem can be formally expressed, the best solution should be found by testing all the values $\theta_X \in]0, 1[^4$ and $\mathbf{\Sigma}_{\mathbf{x}} \in \{\mathbf{0}, \mathbf{1}\}^N$; that exhaustive search is too complex to implement, and the solution would depend on the step adopted to sweep the interval $]0, 1[$ for θ_X. Instead, we make progressively precise estimation of the parameters based on the mean log-likelihood function. Let l be the label of the current iteration of the algorithm. Then, the new estimate θ_X^{l+1} is produced given the previous estimates θ_X^l and $\mathbf{\Sigma}_{\mathbf{x}}^l$.

$$\theta_X^{l+1} = \arg \max_{\theta_X} \mathbb{E}_{\mathbf{\Sigma}_{\mathbf{x}}|\mathbf{X}, \theta_X^l} \log \left(\mathbb{P}_{\theta_X}(\mathbf{x}, \sigma_{\mathbf{x}})\right) \tag{3.5}$$

As side products, we also produce the new estimate $\hat{\mathbf{\Sigma}}_{\mathbf{x}}^{l+1}$ for the current EM iteration.

Since $\log(\cdot)$ is a strictly increasing function, the value $\hat{\theta}_X$ that maximizes $\mathbb{P}_{\theta_X}(\mathbf{x}, \sigma_{\mathbf{x}})$ also maximizes $\log\left(\mathbb{P}_{\theta_X}(\mathbf{x}, \sigma_{\mathbf{x}})\right)$. The final estimates $\hat{\theta}_X$ and $\hat{\mathbf{\Sigma}}_{\mathbf{x}}$ are obtained after an arbitrary number of iterations, when the algorithm has converged, or when a maximum number of iterations has been reached.

The algorithm is composed of *two* steps: the *Expectation* (E-step) and the *Maximization* (M-step). The E-step consists into finding the expression for the mean log-likelihood function, and the M-step consists into updating the parameters given the observed data and the current estimates of the parameters.

3.2.4.1 E-step: Computation of the mean log-likelihood function

We first expand the likelihood function using the Bayes rule:

$$\mathbb{P}_{\theta_X}(\mathbf{x}, \sigma_{\mathbf{x}}) = \mathbb{P}_{\theta_X}(\mathbf{x}|\sigma_{\mathbf{x}})\mathbb{P}_{\theta_X}(\sigma_{\mathbf{x}}) \tag{3.6}$$

Now, given that the source symbol x_n only depends on the current state σ_n, and given that the states are drawn from a Markov process of order one, the expression (3.6) van be expanded to:

$$\mathbb{P}_{\theta_X}(\mathbf{x}, \sigma_{\mathbf{x}}) = \mathbb{P}_{\theta_X}(\sigma_1) \prod_{n=1}^{N} \mathbb{P}_{\theta_X}(x_n|\sigma_n) \prod_{n=2}^{N} \mathbb{P}_{\theta_X}(\sigma_n|\sigma_{n-1}) \tag{3.7}$$

Substituting the expressions given by Equation (3.3) in each term of the product in Equation (3.7), we obtain:

$$\mathbb{P}_{\theta_X}(\mathbf{x}, \sigma_{\mathbf{x}}) = \mathbb{P}_{\theta_X}(\sigma_1) \prod_{n=1}^{N} \prod_{i=\mathbf{0}}^{\mathbf{1}} p_i^{\delta_{\sigma_n=i,x_n=1}} (1-p_i)^{\delta_{\sigma_n=i,x_n=0}} \prod_{n=2}^{N} \prod_{i=\mathbf{0}}^{\mathbf{1}} \prod_{j=\mathbf{0}}^{\mathbf{1}} t_{ij}^{\delta_{\sigma_{n-1}=i,\sigma_n=j}} \tag{3.8}$$

where δ represents the Kroenecker's symbol $\delta_{bool} = \begin{cases} 1, & if\ bool = true \\ 0, & otherwise \end{cases}$.

Taking the logarithm of this expression (3.8), we obtain:

$$\begin{aligned} \log\left(\mathbb{P}_{\theta_X}(\mathbf{x}, \sigma_{\mathbf{x}})\right) = &\log\left(\mathbb{P}_{\theta_X}(\sigma_1)\right) + \sum_{n=1}^{N} \sum_{i=\mathbf{0}}^{\mathbf{1}} \delta_{\sigma_n=i,x_n=1} \log(p_i) + \delta_{\sigma_n=i,x_n=0} \log(1-p_i) \\ &+ \sum_{n=2}^{N} \sum_{i=\mathbf{0}}^{\mathbf{1}} \sum_{j=\mathbf{0}}^{\mathbf{1}} \delta_{\sigma_{n-1}=i,\sigma_n=j} \log(t_{ij}) \end{aligned} \tag{3.9}$$

Now, we take the expectation of the logarithm 3.9, to obtain the mean log-likelihood function $\mathbb{E}_{\mathbf{\Sigma}_{\mathbf{x}}|\mathbf{X},\theta_X^l} \log\left(\mathbb{P}_{\theta_X}(\mathbf{x}, \sigma_{\mathbf{x}})\right)$. Instead of writing down the whole expression, we just notice that we only have to express the expectation of the $\delta's$. $\forall n \in [1, N], \forall i, j \in \{\mathbf{0}, \mathbf{1}\}, \forall k \in \{0, 1\}$

$$\begin{aligned} &\mathbb{E}_{\mathbf{\Sigma}_{\mathbf{x}}|\mathbf{X},\theta_X^l} \delta_{\sigma_n=i,x_n=k} = \delta_{x_n=k} \mathbb{P}_{\theta_X^l}(\Sigma_n = i) \\ &\mathbb{E}_{\mathbf{\Sigma}_{\mathbf{x}}|\mathbf{X},\theta_X^l} \delta_{\sigma_{n-1}=i,\sigma_n=j} = \mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j) \end{aligned} \tag{3.10}$$

3.2.4.2 M-step: Update rules for the parameters

The maximization of the mean log-likelihood function $\mathbb{E}_{\mathbf{\Sigma_x}|\mathbf{X},\theta_X^l} \log\left(\mathbb{P}_{\theta_X}(\mathbf{x},\sigma_\mathbf{x})\right)$ has to be performed under the following constraints. $\forall i,j \in \{\mathbf{0},\mathbf{1}\}$:

$$p_i \in [0,1],\ and\ t_{ij} \in \left]\,0,1\,\right[\ ,\ and\ \sum_{j\in\{\mathbf{0},\mathbf{1}\}} t_{ij} = 1$$

Using Lagrange multipliers, we derive the update rules for the parameters. $\forall i,j \in \{\mathbf{0},\mathbf{1}\}$:

$$\begin{aligned} p_i^{l+1} &= \frac{\sum_{n=1}^{N} \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{x})\delta_{x_n=1}}{\sum_{n=1}^{N} \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{x})} \\ t_{ij}^{l+1} &= \frac{\sum_{n=2}^{N} \mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j|\mathbf{x})}{\sum_{n=2}^{N} \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{x})} \end{aligned} \tag{3.11}$$

Here, we see that the computation for the next parameter θ_X^{l+1} needs the *a posteriori* values of the states probabilities $\mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{x})$ and $\mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j|\mathbf{x})$. Those quantities can be found using a *forward-backward* algorithm which aims at estimating the state sequence σ given the source realization $\mathbf{x}$.

3.2.4.3 Forward-Backward algorithm for the states probabilities

The choice of the forward-backward algorithm is motivated by the Markovian structure of the states $\mathbf{\Sigma}$. Here, the probabilities of $(\Sigma)_{n=1}^N$ are updated according to the values of the current estimate θ_X^l. The aim is to compute:

$$\begin{aligned} \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{x}) &= \frac{\mathbb{P}_{\theta_X^l}(\Sigma_n = i, \mathbf{x})}{\mathbb{P}_{\theta_X^l}(\mathbf{x})} \\ \mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j|\mathbf{x}) &= \frac{\mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j, \mathbf{x})}{\mathbb{P}_{\theta_X^l}(\mathbf{x})} \end{aligned} \tag{3.12}$$

To that end, we decompose the following expression, to retrieve the equations corresponding to the forward-backward recursions:

$$\begin{aligned} \mathbb{P}_{\theta_X^l}(\Sigma_n = i, \mathbf{x}) &= \sum_{j\in\{\mathbf{0},\mathbf{1}\}} \mathbb{P}_{\theta_X^l}(\Sigma_n = i, \Sigma_{n+1} = j, \mathbf{x}) \\ &= \sum_{j\in\{\mathbf{0},\mathbf{1}\}} \alpha_i^n \cdot \gamma_{i,j}^{n,(n+1)} \cdot \beta_j^{(n+1)} \end{aligned} \tag{3.13}$$

The forward-backward algorithm is run on the trellis shown in Fig. 3.4, which states are the same states $\mathbf{0}$ and $\mathbf{1}$ generating the source symbols, with two branches between the states, labeled by the two values $x_n = 0$ and $x_n = 1$. We define

$$\begin{aligned}\gamma_{i,j}^{n,(n+1)} =& \mathbb{P}_{\theta_X^l}(x_n|\Sigma_n = i) \cdot \mathbb{P}_{\theta_X^l}(\Sigma_{n+1} = j|\Sigma_n = i) \\ \alpha_j^n =& \sum_{i\in\{\mathbf{0},\mathbf{1}\}} \alpha_i^{(n-1)} \cdot \gamma_{i,j}^{(n-1),n} \\ \beta_i^n =& \sum_{i\in\{\mathbf{0},\mathbf{1}\}} \gamma_{i,j}^{n,(n+1)} \cdot \beta_j^{(n+1)}\end{aligned} \tag{3.14}$$

where:

- $\gamma_{i,j}^{n,(n+1)}$ is the transition probability between the states i at position n and j at position $(n+1)$.
- α_j^n is the forward probability for the source to be in state j at position n;
- β_i^n is the backward probability for the source to be in state i at position n.

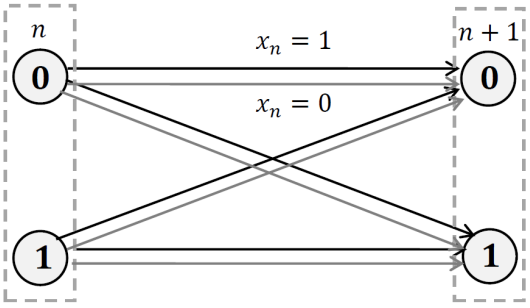

Figure 3.4: Trellis of the Markovian process Σ, on which the *forward-backward* algorithm is run.

Now we define the states APP:

$$\begin{aligned}\mathbb{P}_{\theta_X^l}(\Sigma_n = i, \mathbf{x}) &= \alpha_i^n \cdot \beta_i^n \\ \mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j, \mathbf{x}) &= \alpha_i^{(n-1)} \cdot \gamma_{i,j}^{(n-1),n} \cdot \beta_j^n\end{aligned} \tag{3.15}$$

Normalizing $\mathbb{P}_{\theta_X^l}(\sigma_n, \mathbf{x})$ and $\mathbb{P}_{\theta_X^l}(\sigma_{n-1}, \sigma_n, \mathbf{x})$, we get $\mathbb{P}_{\theta_X^l}(\sigma_n|\mathbf{x})$ and $\mathbb{P}_{\theta_X^l}(\sigma_{n-1}, \sigma_n|\mathbf{x})$.

Decision

$$\forall n \in [1, N] \hat{\sigma}_n = \begin{cases} \mathbf{1}, \ if\ \mathbb{P}_{\theta_X^l}(\sigma_n = 1|\mathbf{x}) \geq 0.5 \\ \mathbf{0}, \ otherwise \end{cases} \tag{3.16}$$

3.2.4.4 Precision of the Baum-Welch algorithm

In this section, we assess the reliability of the Baum-Welch algorithm. We consider a Gilbert-Elliott source, having the parameters $\theta_X = \{t_{\mathbf{10}} = 0.03, t_{\mathbf{01}} = 0.01, p_{\mathbf{1}} = 0.7, p_{\mathbf{0}}\}$ where $p_{\mathbf{0}}$ ranges from 0 to 1. The algorithm is initialized with the values $\theta_0 = \{t_{\mathbf{10}}^0 = 0.1, t_{\mathbf{01}}^0 = 0.1, p_{\mathbf{1}}^0 = 0.51, p_{\mathbf{0}}^0 = 0.49\}$, the parameters are assumed to be well estimated when their variation is under $\frac{1}{N}$, and the maximum number of

iterations is 100. For each value of p_0, 10^4 blocks of length $N = 1584$ are tested (that is the same block length as the video bit planes in the DVC experiments). The results are shown in Fig.3.5.

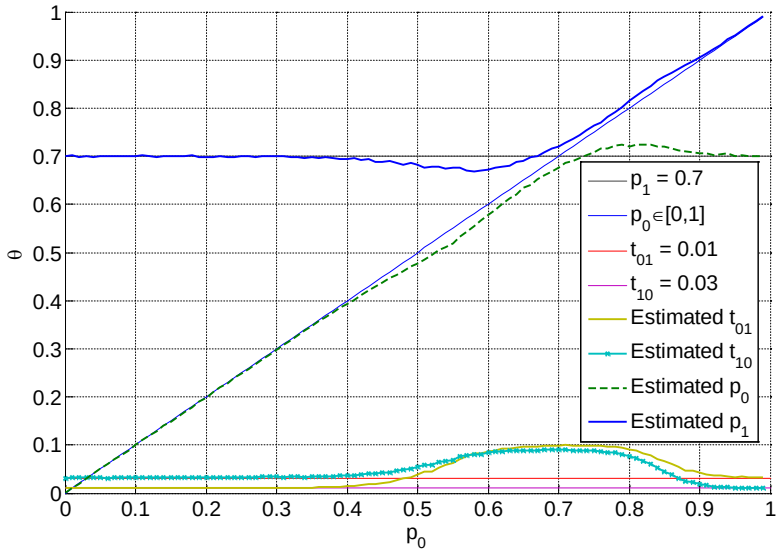

Figure 3.5: Estimated values of the parameters, obtained with the Baum-Welch algorithm for $N = 1584$ when p_0 varies from 0 to 1.

As plotted in Fig. 3.5, the algorithm is stable when p_0 remains low in comparison to p_1 ($p_0 \in]0, 0.4[$) and when p_0 is big in comparison to p_1 ($p_0 \in]0.9, 1[$). The algorithm is less and less able to distinguish the states when their Bernoulli parameters become similar ($p_0 \in [0.4, 0.9]$), so the estimated values are far from the real parameters. In particular, when $p_0 = p_1 = 0.7$, the algorithm cannot distinguish the states and the transitions within the source realization, so the transition probabilities remain at their initialization values ($t_{10} = t_{01} = 0.1$). When $p_0 > p_1$, the roles taken by the two states are permuted: **0** is estimated as **1**, and *vice versa*. This comes from the initial values of the parameters which impose that $p_1 > p_0$.

3.2.4.5 Convergence speed of the Baum-Welch estimator

Here, we want to assess the minimal number of Baum-Welch iterations to obtain acceptably good estimates of the parameters. To that end, we show in Fig. 3.6 the behavior of the algorithm with the iterations. We test the same parameter values $\theta = \{p_0, p_1 = 0.7, t_{10} = 0.03, t_{01} = 0.01\}$ as in Fig. 3.5. The are attributed one color each, and their estimates are attributed the same color in an increasingly stronger tone for more iterations. The results for 5 steps are shown: 5 iterations, 10 iterations, 20 iterations, and 200 iterations.

What is obvious from Fig. 3.6 is the more the states are similar the more iterations are needed to obtain good enough estimates. 10 iterations are large enough when $p_0 \leq 0.05$ and 20 iterations for $p_0 \leq 0.2$. When p_0 is in the close neighborhood of p_1, even 200 iterations are not sufficient for the algorithm to converge to the right values.

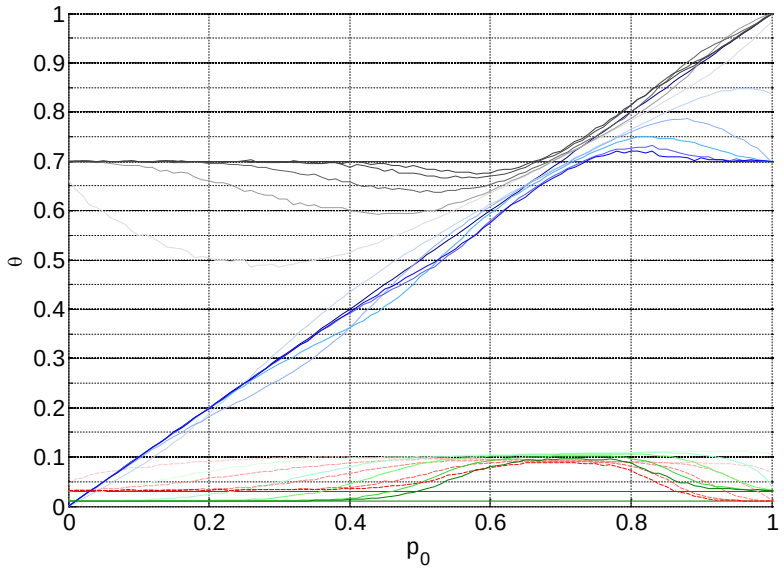

Figure 3.6: Convergence speed of the Baum-Welch algorithm for the estimation of the GE source parameters. The results are shown for 5 iterations, 10 iterations, 20 iterations, and 200 iterations.

In our experimental tests, we decide to test, at each iteration, if the parameters change in a significant way; otherwise, we decide that the algorithm has converged to the final estimates. Practically, we fix the change threshold to be $\frac{1}{N}$ for a source sequence of length N bits, with 100 maximum number of iterations.

3.2.4.6 Influence of the initialization values

In this Section, we justify our choice for the initialization of the Baum-Welch algorithm at the values $\theta^0 = \{p_{\mathbf{0}}^0 = 0.49, p_{\mathbf{1}}^0 = 0.51, t_{\mathbf{10}}^0 = 0.1, t_{\mathbf{01}}^0 = 0.1\}$. To that end, we test several values for $p_{\mathbf{0}}^0$ and $p_{\mathbf{1}}^0$ to estimate the true parameters $\theta = \{p_{\mathbf{0}}, p_{\mathbf{1}} = 0.7, t_{\mathbf{10}} = 0.03, t_{\mathbf{01}} = 0.01\}$ (as the test conditions in Fig. 3.5). As the values of $t_{\mathbf{10}}$ and $t_{\mathbf{01}}$ need to be small to observe persistence of the states, we leave their initial values $t_{\mathbf{10}}^0$ and $t_{\mathbf{01}}^0$ at 0.1. As a principle, we decide that $p_{\mathbf{1}} \geq p\mathbf{0}$, then the same order needs to be followed by their initial values. Therefore, we assess the performance of the estimator for the initial values $\{p_{\mathbf{0}}^0 = 0.1, p_{\mathbf{1}}^0 = 0.9\}$ and $\{p_{\mathbf{0}}^0 = 0.4, p_{\mathbf{1}}^0 = 0.6\}$; the results are respectively shown on the left and the right in Fig. 3.7.

We observe that the two initial values incur very different performances of the same algorithm. However, taking into account the performance in Fig. 3.5, the two initial values tested in Fig. 3.7 are not better ones. That is why we decide to use the initial values $\theta^0 = \{p_{\mathbf{0}}^0 = 0.49, p_{\mathbf{1}}^0 = 0.51, t_{\mathbf{10}}^0 = 0.1, t_{\mathbf{01}}^0 = 0.1\}$ in all our experiments with GE sources.

3.3 On the predictive Binary Symmetric Channel

In the previous Section, we presented the *additive* BSC, in which the noise Z that is produced by the channel is independent of the input X, and the side-information

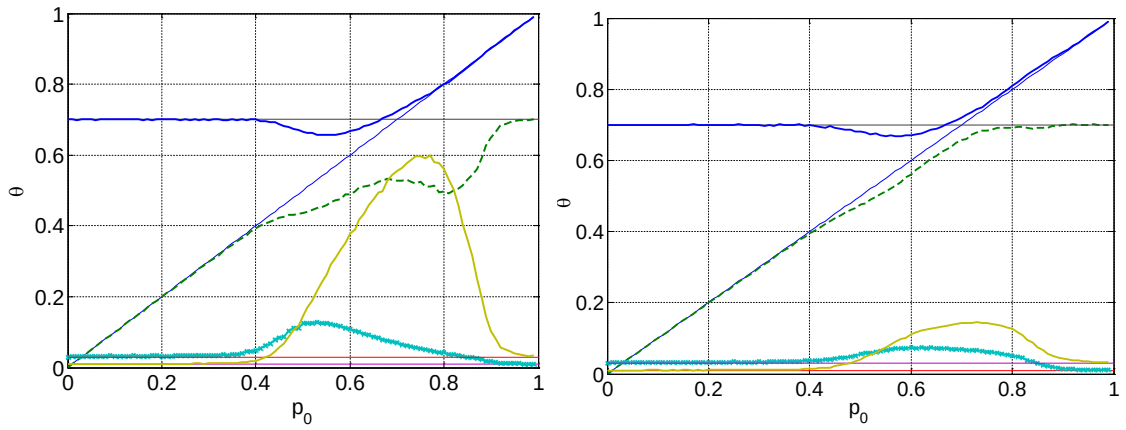

Figure 3.7: Influence of the initial values of the parameters.

is obtained by $Y = X \oplus Z$. We showed that considerable gain was expected when coding the two correlated sources X and Y, with respect to the case where X and Y are uniform. However, when it comes to coding the bit planes produced by a DVC system, the additive correlation model is not always true. Therefore, we introduce a novel correlation model called *"predictive"* correlation model.

3.3.1 Definition of the predictive BSC

Definition 3.2. *An (X, Y, p) predictive BSC is a channel with binary input X, binary output Y. The noise $Z \sim \mathcal{B}(p)$ is independent of the channel output s.t. $X = Y \oplus Z$.*

This model corresponds to the case where Y represents a *prediction* of X. Z is therefore an innovation noise independent of Y. The introduction of the predictive channel is motivated by the DVC application. In this context, X represents the current image to be compressed and Y represents the prediction of X based on previous and future images obtained at the decoder. Therefore, the noise Z is an innovation noise and is more likely to be independent of Y than of X. Unfortunately, for a predictive channel the compression rate for X does not reduce as the non-uniformity of X increases.

In Section 5.2.3, we will show the accuracy of this predictive correlation model for the DVC application.

3.3.2 Achievable coding rate for non-uniform sources

Let $X \sim \mathcal{B}(p_X)$ and $Y \sim \mathcal{B}(p_Y)$ be two correlated Bernoulli sources. When the correlation is modeled as a *predictive* channel, we get the following result for the

distributed asymmetric coding:

Claim 3.2. *Let $X \sim \mathcal{B}(p_X)$ and $Y \sim \mathcal{B}(p_Y)$ be two correlated sources, where the correlation is modeled as a virtual (X, Y, p) predictive BSC. We consider the asymmetric distributed problem, where Y is available at the decoder and compressed at its entropy $H(Y)$ and X is compressed at its conditional entropy $H(X|Y)$.*
The non-uniformity of X does not reduce the compression rate of X.

Proof. Here $H(X|Y) = H(Z)$ since X is dependent of the correlation noise. Therefore, the coding rate for X only depends on the noise statistics, and the non-uniformity of X does not reduce its coding rate. □

For a *predictive* channel, the compression rate of X does not decrease as the non-uniformity of X increases. In the following Lemma 3.3, we show that a mismatch between the true and the assumed correlation models always degrades the performance of the decoder when the true correlation model is *additive*.

Lemma 3.3. *Let $X \sim \mathcal{B}(p_X)$ and $Y \sim \mathcal{B}(p_Y)$ be two correlated sources, where the correlation is modeled as a virtual additive (X, Y, p) BSC. We consider the asymmetric distributed problem, where Y is available at the decoder and compressed at its entropy $H(Y)$, and X is compressed at its conditional entropy $H(X|Y)$.*
A mismatch between the true correlation model and the one assumed by the codec implies a rate loss if the sources are non-uniform.

Proof. Assume that the correlation channel model is *additive*. The lower rate bound is then $H(X|Y) \leq H(Z)$. If the decoding if performed with a *predictive* channel model, then the lower achievable rate is $H(Z)$. A rate loss of $H(Z) - H(X|Y)$ then results from the mismatch. □

When the true correlation model is *predictive*, a decoder mismatch also incurs a rate loss. Actually, the role of the optimal decoder for the predictive correlation model is to find the difference between the correlated sources, regardless of their distribution. This is empirically shown in the experiments Section 4.1.2.2 with synthetic sources.

3.3.3 Achievable coding rate for GE sources

We consider the problem of asymmetric DSC for GE sources X and Y, where Y is transmitted at a rate greater than its entropy-rate $H(\mathcal{Y})$, and is thus available at the decoder; X is transmitted at a rate greater than its conditional entropy-rate $H(\mathcal{X}|\mathcal{Y})$. When the correlation between the sources X and Y is a *predictive* channel,

we claim the following result:

Claim 3.3. *Let $X \sim GE(\theta_X)$ and $Y \sim GE(\theta_Y)$ be two correlated sources, where the correlation is modeled as a virtual (X, Y, p) predictive BSC. We consider the asymmetric DSC problem, where Y is transmitted at a rate greater than its entropy-rate $H(\mathcal{Y})$; and X is transmitted at a rate greater than its conditional entropy-rate $H(\mathcal{X}|\mathcal{Y})$.*
The minimum transmission rate for X is not reduced, with respect to that of a uniform source. More precisely $H(\mathcal{X}|\mathcal{Y}) = H(Z)$.

Proof. Here Z is independent of Y. Therefore, the minimum coding rate for X only depends on the noise statistics. Then the conditional entropy-rate of X is given by $H(\mathcal{X}|\mathcal{Y}) = H(Z)$; the minimum transmission rate for X is not reduced, with respect to that of a uniform source. □

This result generalizes the result in Claim 3.2 for non-uniform sources to the more general case on hidden Markov sources. More generally, exploiting the statistics of the source does not reduce the coding rate of X, in the distributed coding setup, when the correlation channel is *predictive*. The following Lemma 3.4 generalizes the result in Lemma 3.3 for non-uniform sources, on the mismatch between the additive and the predictive correlation channel models, when the true model is *additive*.

Claim 3.4. *Let $X \sim GE(\theta_X)$ and $Y \sim GE(\theta_Y)$ be two correlated memory sources, where the correlation is modeled as a virtual additive (X, Y, p) BSC. We consider the asymmetric DSC problem, where Y is available at the decoder and coded at a rate greater than its entropy-rate $H(\mathcal{Y})$, and X is coded at a rate greater than its conditional entropy-rate $H(\mathcal{X}|\mathcal{Y})$.*
A mismatch between the true correlation model and the one assumed by the decoder always implies a coding rate loss if the sources are not uniform.

Proof. If the decoding if performed with a predictive correlation channel model while the true channel is additive, then the true minimum achievable rate is $H(\mathcal{X}|\mathcal{Y}) = H(Z) - [H(\mathcal{Y}) - H(\mathcal{X})]$; a coding rate loss is thus incurred by the decoder with the predictive assumption since $H(\mathcal{Y}) - H(\mathcal{X}) \geq 0$. □

Similarly to non-uniform sources, when the true correlation model is *predictive*, a decoder mismatch also incurs a rate loss, this is empirically shown in the experiments Section 4.2.8 with synthetic sources.

Now that we have presented the two source models, i.e. *non-uniform* and *GE* sources, as well as the two correlation channel models, i.e. *additive* and *predictive* BSC, we turn to estimating the BSC parameter p. Our method is valid regardless of the source distribution or the channel model.

3.4 Fast and optimal BSC parameter estimation

In this Section, we propose a novel estimation of the correlation channel parameter, when the channel is a BSC. This estimation is simple and can be computed prior to decoding. It exploits the information from the syndrome $\mathbf{s_x}$, the side-information $\mathbf{y}$, and the parity-check matrix $\mathbf{H}$. The estimator uses the *probability* of occurrence of *ones* in the sum of $\mathbf{s_x}$ and $\mathbf{s_y}$. We show that this probability is a bijective function of p, and we derive the *Maximum Likelihood* (ML) estimator for p with respect to $\mathbf{s_x}$ and $\mathbf{s_y}$. The estimation is performed prior to decoding, which makes it independent of any decoding algorithm. We propose a second estimator of p with respect to $\mathbf{s_x}$ and $\mathbf{y}$ which requires the use of an *Expectation-Maximization* (EM) algorithm. We show that the EM only slightly improves the previous estimate, and we conclude that the first method already provides a good estimate. The proposed methods can be applied to the BSC parameter estimation for channel coding.

Let $Z \sim \mathcal{B}(p)$, $p \in [0, 0.5]$, represent the correlation channel between two arbitrary binary sources X and Y. Let $\mathbf{x}$ and $\mathbf{y}$, of length N, be two vectors of the respective realizations of X and Y. $\mathbf{x}$ and $\mathbf{y}$ differ by some noise $\mathbf{z}$, which is the realization of Z. Here, the BSC can be additive or predictive. Let $\mathbf{H}$ designate the parity check matrix of an (N, K) linear block code $\mathcal{C}$, of size $(N-K) \times N$. Let H_m designate the m-th row of $\mathbf{H}$. Let $\mathbf{s_x} = \mathbf{Hx}$, of length $(N-K)$, be the syndrome of $\mathbf{x}$ and $\mathbf{s_y} = \mathbf{Hy}$ of length $(N-K)$ the syndrome of $\mathbf{y}$.

3.4.1 ML estimator of p with respect to the syndromes

The Turbo and LDPC decoders detailed in Sections 1.3.1.3 and 1.3.2.5 respectively are fed with the syndrome $\mathbf{s_x}$ and the side-information $\mathbf{y}$. In the following, we derive an estimator for p based on the knowledge of the syndromes $\mathbf{s_x}$ and $\mathbf{s_y}$ only, and we show that our estimate is efficient for a large range of values of p (since it is the ML estimator) and can be performed prior to decoding. We first consider regular block codes.

3.4.1.1 ML estimator for regular block codes

Regular block codes have the same number of *ones*, noted d_c, in every row of $\mathbf{H}$. d_c is also called *"syndrome degree"*. We first enunciate a lemma that characterizes the Bernoulli nature of the syndrome of $\mathbf{z}$.

Lemma 3.4. *Let* $\mathbf{H}$ *be the matrix of a linear block code in which all the rows are linearly independent and contain the same number of ones* (d_c)*. Let* $\mathbf{z}$ *be the realization of* $Z \sim \mathcal{B}(p), p \in [0, 0.5]$*. Let* $\mathbf{s_z} = \mathbf{Hz}$*.*
The syndrome $\mathbf{s_z}$ *can be seen as the realization of a Bernoulli random variable* S_Z *of parameter* q*, s.t.*

$$q(p) = \sum_{\substack{i=1 \\ i \ odd}}^{d_c} p^i (1-p)^{d_c - i} \binom{d_c}{i} \tag{3.17}$$

Proof. First, since the rows of $\mathbf{H}$ are linearly independent, the syndrome symbols are independent. Now, let s_{zm} be the m^{th} element of $\mathbf{s_z}$ and let q be the probability of s_{zm} being a *one*. Since Z is an i.i.d. process of law $\mathcal{B}(p)$, and since s_{zm} is the sum of d_c elements of Z, $q(p)$ is given by Equation (3.17) and it is the same for all the symbols $(s_{zm})_{m=1}^{(N-K)}$. Therefore, the syndrome symbols are independent and identically distributed realizations of an (*iid*) binary source, i.e. a Bernoulli source, noted S_Z, of parameter q. □

Now, that we have characterized the process S_Z, we can derive an estimator of its Bernoulli parameter q. This is stated in the following corollary.

Corollary 3.1. *The ML estimator of q with respect to $\mathbf{s_z}$ is the estimate $\hat{q}$ of the mean of the i.i.d. Bernoulli process S_Z.*

$$\hat{q} = \frac{1}{N-K} \sum_{m=1}^{N-K} s_{zm} \tag{3.18}$$

Proof. Since $\mathbf{s_z}$ is the realization of a Bernoulli variable $S \sim \mathcal{B}(q)$ (from Lemma 3.4), $\sum_{m=1}^{N-K} s_{zm}$ is a sufficient statistic for the estimation of q with respect to $\mathbf{s_z}$, and the mean of S_Z, in (3.18), is the ML estimator of q. □

Now, we turn to the DSC problem, where $\mathbf{s_z}$ is not observable, but only $\mathbf{s_x}$ and $\mathbf{y}$; the following Theorem 3.1 gives the ML estimator of p with respect to the available data.

Theorem 3.1. *Let $\mathbf{H}$ be the matrix of a linear block code in which all the rows are linearly independent and contain the same number of ones. Let $\mathbf{x}$ be the realization of the binary source X, and let $\mathbf{s_x} = \mathbf{Hx}$ be its syndrome. Let Y be another binary source which is correlated to X in the following manner: $\exists Z \sim \mathcal{B}(p)$ s.t. $p \in [0, 0.5]$ and $Y = X \oplus Z$. Let $\mathbf{y}$ be a realization of Y, and let $\mathbf{s_y}$ be its syndrome. Let $f : p \to q(p)$, where $q(p)$ is given in (3.17).*
The Maximum Likelihood estimator for p with respect to $(\mathbf{s_x}, \mathbf{s_y})$ is:

$$\hat{p} = f^{-1}(\hat{q}) \tag{3.19}$$

where $\hat{q}$ is given in (3.18), and f^{-1} is the inverse of f.

Proof. Let $\mathbf{s_x} = (s_{xm})_{m=1}^{(N-K)}$ and $\mathbf{s_y} = (s_{ym})_{m=1}^{(N-K)}$. The joint probability of $\mathbf{s_x}$ and $\mathbf{s_y}$ can be factored as:

$$\begin{aligned} \mathbb{P}(\mathbf{s_x}, \mathbf{s_y}) &= \mathbb{P}(\mathbf{s_x}) \cdot \mathbb{P}(\mathbf{s_x} \oplus \mathbf{s_y}) \\ &= \mathbb{P}(\mathbf{s_x}) \cdot q^{\left(\sum_{m=1}^{N-K} s_{xm} \oplus s_{ym}\right)} (1-q)^{\left((N-K) - \sum_{m=1}^{N-K} s_{xm} \oplus s_{ym}\right)} \end{aligned} \tag{3.20}$$

From [Kay93, Theorem 5.1], $\sum_{m=1}^{N-K} s_{xm} \oplus s_{ym}$ is a sufficient statistic for the estimation of q. The ML estimator of q, with respect to $(\mathbf{s_x}, \mathbf{s_y})$, is thus $\hat{q} = \frac{1}{N-K} \sum_{m=1}^{N-K} s_{xm} \oplus s_{ym}$. We denote $f(p) = q(p)$ for clarity of notation.

f is a strictly increasing one-to-one function of p in $[0, 0.5]$, and we denote f^{-1} its inverse, s.t. $p = f^{-1}(q)$. It follows from [Kay93, Theorem 7.2], that the ML estimator of p, with respect to $(\mathbf{s_x}, \mathbf{s_y})$, is $\hat{p} = f^{-1}(\hat{q})$. □

This ML estimator $\hat{p}$ does not depend on the distribution of X and Y since $\mathbf{s_x} \oplus \mathbf{s_y}$ only depends on the BSC modeling their correlation. There is no analytical expression of the inverse function f^{-1}; the inversion can be implemented numerically (with a correspondence table for example). Note that our estimator is biased since f is not a linear function.

3.4.1.2 ML estimator for irregular block codes

We now derive the ML estimator of p when the syndrome symbols have different degrees. Irregular LDPC codes are characterized by their *variable degree distribution* $\Lambda(x)$ which is the distribution of the number of *ones* in the columns of $\mathbf{H}$, and by their *check degree distribution* $\Phi(x)$ which is the distribution of the number of *ones* in the rows of $\mathbf{H}$. $\forall m \in [1, (N-K)]$, let d_{cm} be the degree of s_{zm}. Let d^{max} be the maximum check degree. Let $f_{d_{cm}}(p) = \sum_{\binom{i=1}{i\ odd}}^{d_{cm}} p^i (1-p)^{d_{cm}-i} \binom{d_{cm}}{i}$.

We want to find the ML estimate $\hat{p}$ with respect to $\mathbf{s_x}$ and $\mathbf{s_y}$. It is the solution of the maximization problem:

$$\hat{p} = \arg\max_p \mathbb{P}\left(\mathbf{s_x}, \mathbf{s_y}\right)(p)$$

where the joint probability of the syndromes symbols can be expressed in function of p, as:

$$\mathbb{P}\left(\mathbf{s_x}, \mathbf{s_y}\right)(p) = \prod_{m=1}^{N-K} f_{d_{cm}}(p)^{s_{xm} \oplus s_{ym}} \left(1 - f_{d_{cm}}(p)\right)^{1 - s_{xm} \oplus s_{ym}} \tag{3.21}$$

Since $\log(\cdot)$ is a strictly increasing function, the value of $\hat{p}$ that maximizes $\mathbb{P}\left(\mathbf{s_x}, \mathbf{s_y}\right)(p)$, in (3.21), also maximizes $\log\left(\mathbb{P}\left(\mathbf{s_x}, \mathbf{s_y}\right)(p)\right)$. In the expression (3.22), the sum on the syndrome symbols has been re-organized in order to group the syndrome symbols having the same degrees.

$$\frac{d}{dp}\left(\log\left(\mathbb{P}\left(S_1^N\right)(p)\right)\right) = \sum_{j=1}^{d^{max}} \sigma_j \frac{f_j'(p)}{f_j(p)} - (1-\sigma_j)\frac{f_j'(p)}{1-f_j(p)} \tag{3.22}$$

where, $\forall j \in [1, d^{max}]$, $\sigma_j = \frac{1}{N_j} \sum_{\binom{m=1}{d_{cm}=j}}^{N-K} s_{zm}$, and $f_j'(p) = \frac{d}{dp} f_j(p)$, and N_j is the number of symbols with degree j. The ML estimator $\hat{p}$ is found by zeroing the derivative (3.22) since it can be shown to satisfy the constraints $p \in [0, 0.5]$.

The method can be applied to the parameter estimation for *channel coding* over the BSC, since the channel decoder is a particular case of the DSC decoder, with $\mathbf{s_x} = \mathbf{0}$ (See Section 3.4.4.2).

3.4.2 Improved estimation of p using an EM algorithm

The ML estimator presented in Section 3.4.1 only uses the information from $\mathbf{s_x}$ and $\mathbf{s_y}$, which is not optimal since the information from $\mathbf{y}$ is not fully exploited. In this Section, an EM algorithm [Rab89] is used to improve the estimator $\hat{p}$. The EM is an optimization procedure that updates $\hat{p}$ through the decoding iterations, the convergence is acquired since it improves the estimator likelihood at each iteration [Wu83]. Let l be the label of the current decoding iteration, and p^l be the current estimate. Then the next estimate is the solution to the maximization problem:

$$p^{(l+1)} = \arg\max_{p} \left(\mathbb{E}_{\mathbf{X}|\mathbf{S_X},\mathbf{Y},p^l} \Big[\log \left(\mathbb{P}_p(\mathbf{s_x}, \mathbf{y}, \mathbf{x}) \right) \Big] \right) \tag{3.23}$$

and is

$$p^{(l+1)} = \frac{1}{N} \sum_{n=1}^{N} |y_n - \mathbb{P}_n| \tag{3.24}$$

where $\mathbb{P}_n$ is the *a posteriori* probability $\mathbb{P}_{p^l}(X_n = 1|\mathbf{s_x}, \mathbf{y})$. This update rule is the same as presented in [GFZ01, Equation 3]. However, our EM algorithm differs from [GFZ01] since it is initialized with our efficient estimator $p^0 = f^{-1}(\hat{q})$, see (3.19).

In Section 3.1.3, we recalled the MVU parameter estimator (Equation (3.2)) for the source X, knowing its realization $\mathbf{x}$. If only the *syndrome* of the source is known, the ML parameter estimator presented in this Section for the BSC can also be performed to find p_X, provided that $H(p_X)$ is lower than the code rate.

3.4.3 Simulation results

3.4.3.1 Precision of the estimator

We implement a DSC system using two LDPC codes of rates 0.5 and 0.7, which has the *variable degree distribution* $\Lambda(x) = 0.483949x + 0.294428x^2 + 0.085134x^5 + 0.074055x^6 + 0.062433x^{19}$ and the *check degree distribution* $\Phi(x) = 0.741935x^7 + 0.258065x^8$. Each row of the two matrices $\mathbf{H}$ is linearly independent of one another. For each code rate, we consider two codes of respective lengths $N = 1000$ and $N = 10000$.

The sources are uniform Bernoulli, and we test the estimation for different values of p. We show the means of the initial estimators $\hat{p}$ in Fig. 3.8 for the codes of size $N = 1000$; we also show the parameters estimated by the EM algorithm, using the first estimate as an initialization value. The parameter is updated, according to (3.24), each 20 iterations of the decoding to observe convergence. Similarly, the results for $N = 10000$ are shown in Fig. 3.9.

We note from Fig. 3.8 and Fig. 3.9 that the performance of the estimators improve with the code length and with the code rate. To have a better view on the performance of the estimators, we show in Fig. 3.10 the biases of the estimators for code length $N = 1000$, and for the two code rates 0.5 and 0.7. Similarly, we show in Fig. 3.11 the biases of the estimators for code length $N = 10000$, and for the two code rates 0.5 and 0.7.

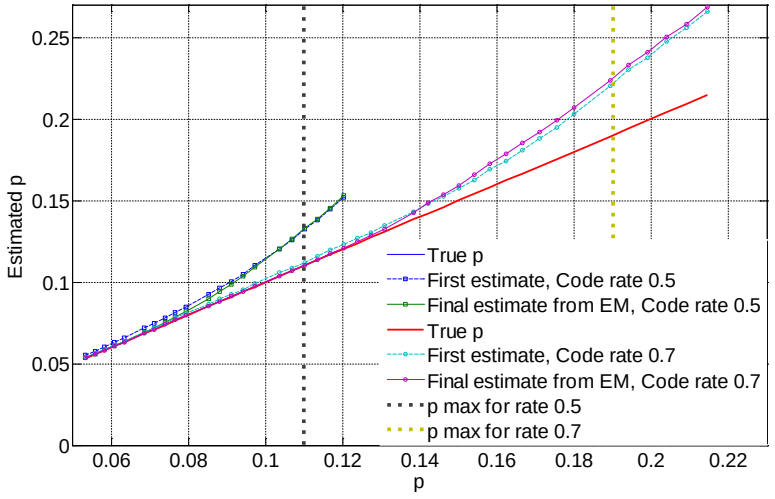

Figure 3.8: Means of the estimated $\hat{p}$ for the two codes of lengths $N = 1000$, for code rates 0.5 and 0.7.

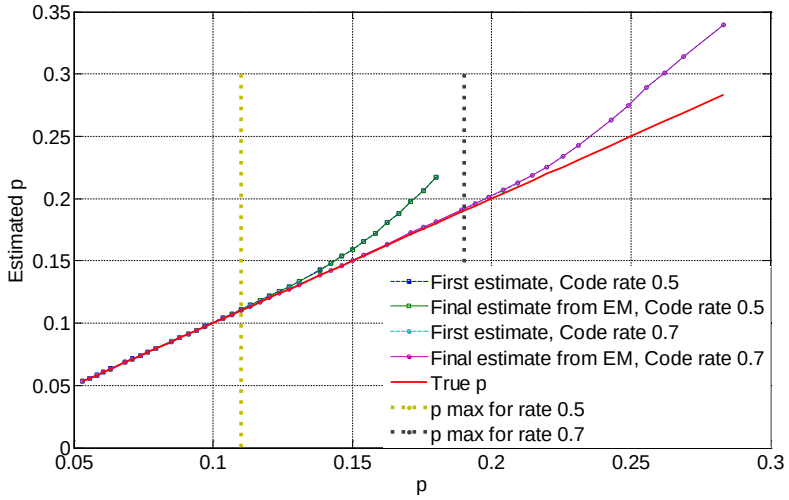

Figure 3.9: Means of the estimated $\hat{p}$ for the two codes of lengths $N = 10000$, for code rates 0.5 and 0.7.

The means of the estimated parameters are very close to the true values (the biases are small). The estimator performance improves with the code length; the gap to the true parameter can go under 10^{-3} for $N = 1000$, and it is under 10^{-4} for $N = 10000$. The EM does not improve the performance of the estimator in a significant amount; this means that the initial estimate of the parameter is already very good. Note that we could assess the performance of the estimator for the whole range of values from 0 to 0.5, but we restrain to $p \leq 0.11$ for code rate 0.5 (resp. to $p \leq 0.19$ for code rate 0.7) because of the LDPC code we use; actually, for the SW setup, the maximum value of p that can realistically be "corrected" by the code verifies $H(p) = 0.5$ (resp. $H(p) = 0.7$), *i.e.* $p = 0.11$ (resp. $p = 0.19$) for the code of rate 0.5 (resp. 0.7). For efficient estimation of higher values of p, a code allowing lower compression rates must be used.

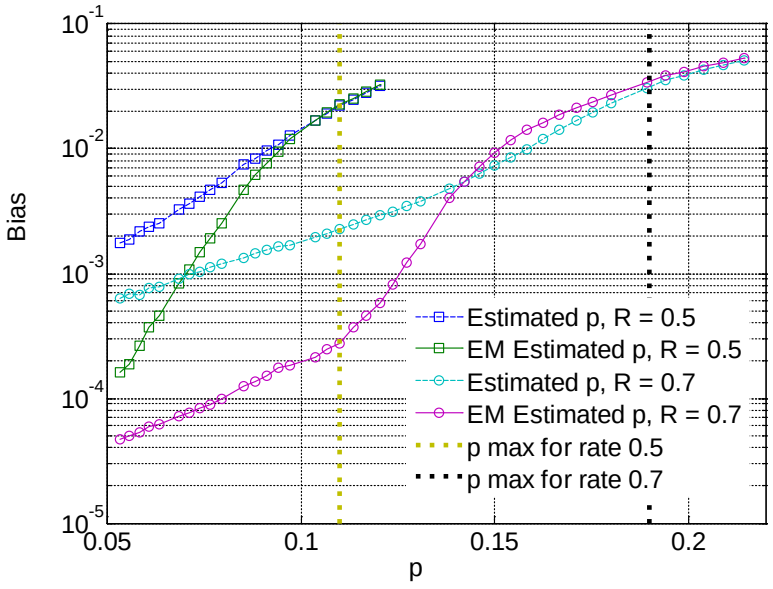

Figure 3.10: Biases of the estimated $\hat{p}$ for the two codes of rates 0.5 and 0.7, for code length $N = 1000$.

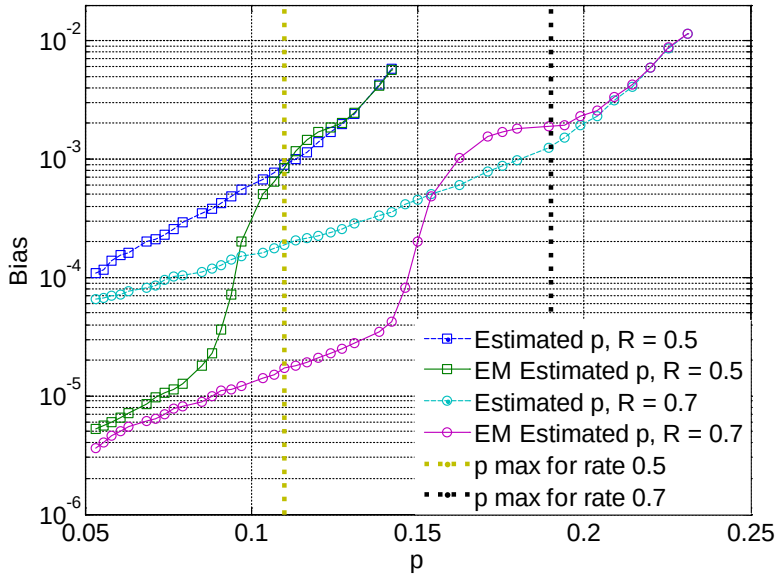

Figure 3.11: Biases of the estimated $\hat{p}$ for the two codes of rates 0.5 and 0.7, for code length $N = 10000$.

3.4.3.2 Distributed Source Coding using the estimated parameters

The performance of the decoder using the initial estimated parameter from (3.22) is compared to the performance of the genie-aided decoder using the true parameter for code rate 0.5. Both decoders iterate at most 100 times. The results presented in Fig. 3.12 indicate that no degradation at all is observable when using the estimated parameter or the true parameter. The BER of the decoder using the EM algorithm (3.24) is not shown in Fig. 3.12 since no rate loss is observable either.

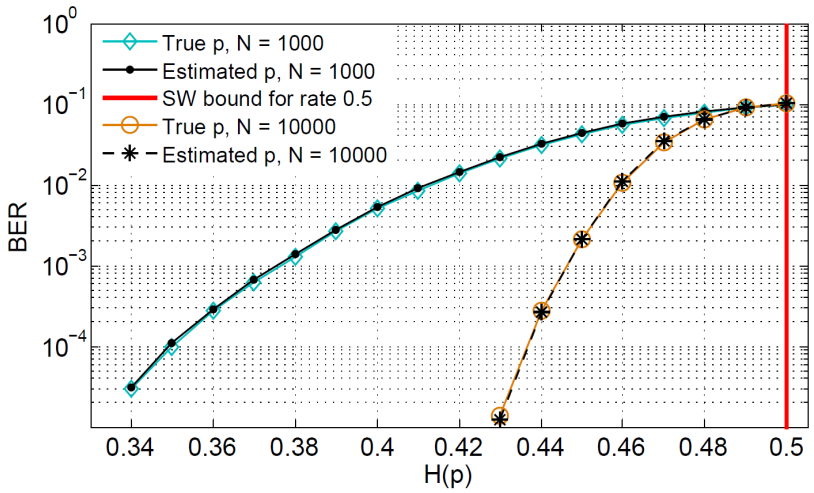

Figure 3.12: Comparison of the BER of X for the genie-aided decoder and the proposed decoder, for N={1000,10000}.

Note that the average bias shown in Fig. 3.8 and Fig. 3.9 is always positive. This is most important since a negative bias would prevent the DSC decoder, using the estimated value $\hat{p}$, from finding the best $\hat{\mathbf{x}}$. More precisely, remember that the decoder finds $\hat{\mathbf{x}}$ as the vector having syndrome $\mathbf{s}_{\mathbf{x}}$ which is at "distance" p to $\mathbf{y}$. Here the distance is the number of different positions in $\mathbf{x}$ and $\mathbf{y}$ divided by N. If the bias of the estimator is negative, then the decoder would not search $\hat{\mathbf{x}}$ in a large enough distance to $\mathbf{y}$.

3.4.3.3 Behavior of the estimator with the block length and the code degree distribution

Now, we investigate on the impact of the degree distribution on the precision of the initial parameter estimation. To that end, we consider the DSC of X and Y using three different degree distributions for the LDPC codes of rate 0.5.

- ⓐ The previous code, that yields the results in Fig. 3.8 and Fig. 3.9;
- ⓑ $\Lambda(x) = 0.488047x + 0.299593x^2 + 0.000420x^4 + 0.034791x^5 + 0.125283x^6 + 0.051865x^{19}$, $\Phi(x) = x^7$;
- ⓒ $\Lambda(x) = 0.547789x + 0.063512x^2 + 0.388698x^3$, $\Phi(x) = 0.318181x^4 + 0.681818x^5$.

The average biases to the true parameters and the variances of the parameter estimations are presented in Table 3.1 for the three codes.

From the results in Table 3.1, it is clear that the precision is better when the syndrome degree is smaller. It is worth noting that, for the DSC setup, the gap to the true parameters is small enough to ensure that the distance to the Slepian-Wolf bound that is achieved using the true parameters is not impaired when using the estimated ones (see Fig. 3.12: the BER of the two decoders for $N = 1000$ are superposed together, as well as the BER of the two decoders for $N = 10000$).

Code	N	Bias	Variance
ⓐ	1000	$2.71 \cdot 10^{-3}$	$4.22 \cdot 10^{-4}$
	10000	$7.61 \cdot 10^{-4}$	$3.39 \cdot 10^{-5}$
ⓑ	1000	$2.55 \cdot 10^{-3}$	$4.22 \cdot 10^{-4}$
	10000	$2.22 \cdot 10^{-4}$	$2.39 \cdot 10^{-5}$
ⓒ	1000	$1.24 \cdot 10^{-3}$	$1.47 \cdot 10^{-4}$
	10000	$9.91 \cdot 10^{-5}$	$1.33 \cdot 10^{-5}$

Table 3.1: Average variances and biases of the initial estimator, for $p \in [0.05, 0.11]$

3.4.3.4 Performance comparison with respect to the Cramer-Rao lower bound

We compare the performances of our estimators, in terms of their *Mean Square Error* (MSE), with the *Cramer-Rao Lower Bound* (CRLB), which is the lower bound on the estimator's Mean Square Error (MSE). For our problem, a bound of the CRLB can be derived by considering the MVU estimator of p knowing the realization $\mathbf{z}$ of length N, of the process $Z \sim \mathcal{B}(p)$. This MVU estimator, that achieves the CRLB is given by:

$$\hat{p} = \frac{1}{N} \sum_{n=1}^{N} z_n \tag{3.25}$$

The MSE of the estimator (3.25), which is the CRLB for this estimation problem, is $\frac{p(1-p)}{N}$. We show in Fig. 3.13 the MSE of the *two* estimators, for $N = 1000$ and for two code rates 0.5 and 0.7. Similarly, the MSE of the *two* estimators, for $N = 10000$ and for two code rates 0.5 and 0.7 are also presented in Fig. 3.14. We also show the CRLB corresponding to each value of p.

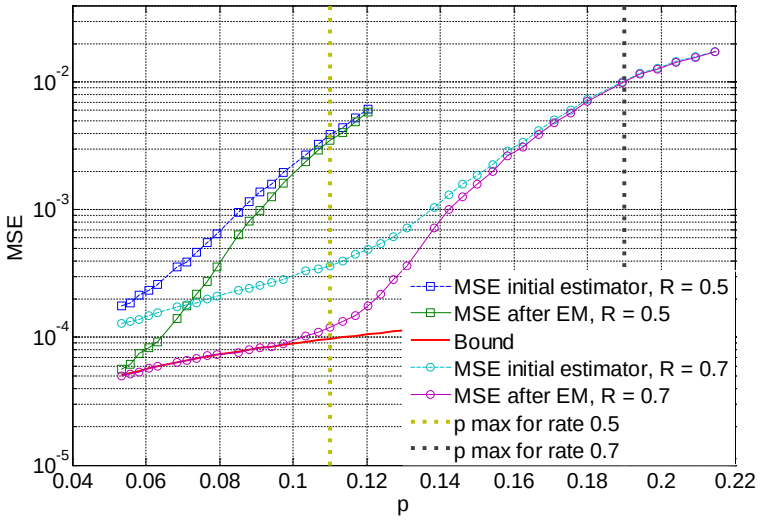

Figure 3.13: Comparison of the MSE of our estimators with the CRLB for each value of p for $N = 1000$.

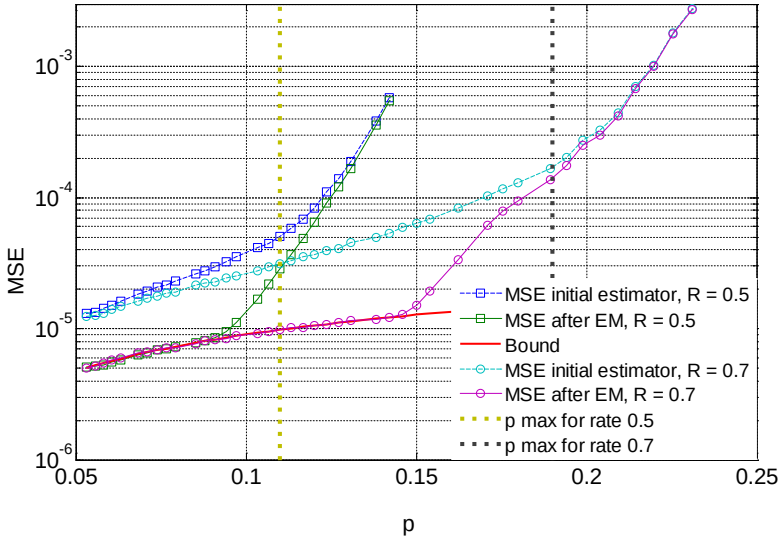

Figure 3.14: Comparison of the MSE of our estimators with the CRLB for each value of p for $N = 10000$.

As seen in Fig. 3.13 and Fig. 3.14, the MSE improves with the block length. The final gap to the CRLB for $N = 10000$ and for $N = 1000$ is very tiny, when the EM algorithm is performed: it is under 10^{-6} when $p \leq 0.09$ for $N = 10000$. When p increases, the gap increases as well, but note that it remains small, with respect to results presented in [ZRS07] where the results after the EM remain far from the CRLB for the codes of rates 0.5 and 0.7.

3.4.4 Discussion

3.4.4.1 Optimality of the cost function "f"

The performance of the estimation method that we proposed closely depends on the check degree distribution $\Phi(x)$. More precisely, for a regular code, Fig. 3.15 presents the behavior of the function $q(p)$ (3.17), for different values of d_c, and for $p \in]0, 0.5]$.

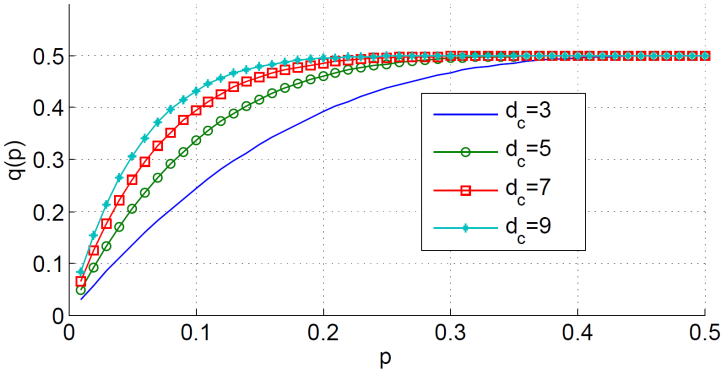

Figure 3.15: Behavior of $q(p)$ for different values of d_c.

We see that the gradient of $q(p)$ increases with d_c. But, as the method directly

exploits the one-to-one correspondence between p and q, the greater the gradient, the less the function is linear, and the less the resulting $\hat{p}$ is precise. Clearly, the estimation method is more efficient when d_c is lower (see Table 3.1); so it seems interesting to use a block code having lower check degrees only.

However, one must take into account that the performance of the code (in terms of reaching the SW bound) also closely depends on the check degree distribution. The *tendency* (in the sense that d_c must remain relatively low for the matrix to remain *sparse*) is that the DSC code is more efficient when d_c is *large*. Therefore, there is a trade-off to be taken into account between estimation performance and DSC performance. This trade-off also depends on the target applications for our estimator. Here, we chose to use the "best" DSC code; that choice is justified by the good performance observed for the 1000 and 10000-long LDPC codes that we use.

3.4.4.2 Extension to BSC parameter estimation for channel coding

Let us consider the channel coding of binary sources. Let $\mathbf{x}$ be a valid *codeword* that is sent over the BSC represented by the signal $Z \sim \mathcal{B}(p)$, with noise realization $\mathbf{z}$. The output of the channel is $\mathbf{y} = \mathbf{x} \oplus \mathbf{z}$. The decoder uses the parity-check matrix $\mathbf{H}$, and $\mathbf{y}$, to estimate the original sequence $\mathbf{x}$, exploiting the fact that $\mathbf{Hx} = \mathbf{0}$. Note that $\mathbf{Hy} = \mathbf{H} \cdot (\mathbf{x} \oplus \mathbf{z}) = \mathbf{Hx} \oplus \mathbf{Hz} = \mathbf{Hz}$; this means that the received sequence $\mathbf{y}$ and the error pattern have the same syndrome $\mathbf{s}$. Using the same argument as for the SW problem (section 3.4.1), the estimator of the mean of the Bernoulli variable, $\hat{q}$, estimated from the syndrome $\mathbf{s}$ can be found with the ML estimator, as in Equation (3.18); and this leads to the ML estimation of the parameter p, as in Equation (3.19). This initial estimate can also be refined using the EM algorithm, throughout the channel decoding iterations.

3.5 Conclusion

In this Chapter, we presented two source models, namely *non-uniform* sources and *GE* sources, for which we designed the appropriate parameter estimators, given the source realization, based on the EM algorithm. We also exhibit the rate gain for their coding, provided that the channel model of their correlation is *additive*. Afterward, we made the difference between the additive and the *predictive* channel models, motivated by the fact that a mismatch between the two models brings non negligible rate loss to the SW coding system. Finally, we presented a fast and optimal method to estimate the BSC parameter using syndrome based DSC codes. The method yields an estimator that performs very close to the CRLB.

Chapter 4

Tools for Distributed Source Coding

This chapter is dedicated to the design of DSC codes for both the asymmetric and the non-asymmetric SW problems. For the *asymmetric* SW setup, we design DSC codes that are able to exploit the non-uniformity (Section4.1) and the memory (Section 4.2) of the sources. Joint decoding and source parameter estimation is performed. Then, we deal with *non-asymmetric* SW setup and present DSC codes that are able to reach any point of the SW rate region; Low-Density Parity-Check (LDPC) codes (Section 4.3.1) as well as Turbo codes (Section 4.3.2) are investigated. The *error propagation* phenomenon, for the coding of uniform Bernoulli sources, is studied, and necessary and sufficient conditions are stated to limit and even to avoid such phenomenon. The results and the conditions are finally extended to non-asymmetric coding of non-uniform sources in Section 4.3.3.

4.1 Asymmetric SW coding of non-uniform sources

Motivated by the compression gain when exploiting the non-uniformity of the source (see Section 3.1), we turn to adapting the existing Slepian-Wolf coding schemes to take into account the knowledge of the source distribution. In Section 4.1.1, we investigate the SW coding of non-uniformly distributed Bernoulli sources using Turbo codes, and we compare the performance of the codec with that of a SW codec based on source codes, namely *distributed quasi-arithmetic codes*; both codes are supposed to have *a priori* knowledge of the source distribution. In Section 4.1.2, we turn to the SW coding of non-uniform sources using LDPC codes, when the source distribution is unknown at the decoder; joint estimation-decoding of the source and its parameter is then performed.

4.1.1 Exploiting the source distribution using Turbo codes

Let X and Y be two binary correlated sources, of respective probabilities $\mathbb{P}(X = 1) = p_X$ and $\mathbb{P}(Y = 1) = p_Y$. The correlation between X and Y is modeled as an additive Binary Symmetric Channel (BSC) of cross-over probability p s.t. $\exists Z \sim \mathcal{B}(p) : Y = X \oplus Z$. Let $\mathbf{s_x}$ be the syndrome of each N-long realization of the source, noted $\mathbf{x} = x_1^N$, and let $\mathbf{y} = y_1^N$ be the realization of the source Y that is correlated to $\mathbf{x}$. The Turbo code we use in this system is composed of two identical

Convolutional codes, and the decoding is performed using the Turbo syndrome trellis first described in [RLG07]. Given the matrix representation of the Turbo code, noted $\mathbf{H}$, the syndrome can be obtained by $\mathbf{s_x} = \mathbf{Hx}$. More details on how to construct syndrome-based Turbo codes can be found in Section 1.3.1.

4.1.1.1 Exploiting the source non-uniformity with Convolutional codes

The Convolutional decoder, knowing the syndrome $\mathbf{s_x}$, the side-information $\mathbf{y}$, and the parameter p_X, looks for the sequence having that syndrome and that non-uniformity which is the closest to $\mathbf{y}$, in terms of their Hamming distance. A modified BCJR is used to take into account the *a priori* information about the non-uniformity. The recurrences for the calculation of the forward state metric, noted α in the literature, and the backward state metric, β, are the same as in Section 1.3.1.3; the only change to bring is the calculation of the branch metric, γ.

Let $(m_t)_{t=1\ldots\tau}$ the sequence of states of the trellis corresponding to a given block $\mathbf{x}$, where τ is the number of trellis slices. Let ν_1^n be the n input bits corresponding to each slice of the trellis, and σ_1^{n-k} the $(n-k)$ output bits labeling the transition between the states m_{t-1} and m_t. Let y_1^n be the current side information bits and s_1^{n-k} current syndrome bits. Let p_j be the extrinsic probability $\mathbb{P}(\hat{x}_j = 1)$. By definition, $\gamma = \mathbb{P}(m_t, \mathbf{y}_t | m_{t-1})$ is computed as:

$$\begin{aligned}\gamma = \delta_{\sigma_1^{n-k}=s_1^{n-k}} \cdot \prod_{j=1}^{n} \Big(& p^{\delta_{\nu_j \neq y_j}} \cdot (1-p)^{\delta_{\nu_j = y_j}} \\ & \cdot p_j^{\delta_{\nu_j=1}} \cdot (1-p_j)^{\delta_{\nu_j=0}} \\ & \cdot p_X^{\delta_{\nu_j=1}} \cdot (1-p_X)^{\delta_{\nu_j=0}} \Big)\end{aligned} \tag{4.1}$$

where δ is the Kronecker's symbol ($\delta_{bool} = 1$ if $bool = true$ and 0 otherwise). The first line in the product of Equation (4.1) formalizes the information from the side information, the second line exploits the extrinsic probabilities and the last line exploits the source probabilities. That last term effectively favors the transitions which are labeled by inputs ν_1^n which distribution are statistically close to the source distribution. Note that for the predictive BSC, there is no need to multiply by the term $p_X^{\delta_{\nu_j=1}} \cdot (1-p_X)^{\delta_{\nu_j=0}}$, since the distribution of the source is not exploited.

4.1.1.2 The Turbo syndrome framework for coding of non uniform sources

The source $X \sim \mathcal{B}(p_X)$, having realizations $\mathbf{x}$ of length N, is mapped into its two syndromes $\mathbf{s_{x1}}$ and $\mathbf{s_{x2}}$, of length $(N-K)$ each. Then the modified BCJR is used for each Convolutional decoder to estimate $\hat{\mathbf{x}}$, passing iteratively updated soft extrinsic messages between them at each iteration. The Turbo decoding stops when $\hat{\mathbf{x}}$ matches the two syndromes, or when a maximum number of iterations is reached.

4.1.1.3 Distributed Arithmetic Coding of non-uniform sources

The work that is presented in this Section includes significant contributions from Dr Enrico Magli and Dr Gabriella Olmo, published in the collaborative paper

[TZMRO09].

Distributed Arithmetic Coding (DAC): encoding module

Let $X \sim \mathcal{B}(p_X)$ be a binary memoryless source emitting realization symbols $\mathbf{x} = (x_n)_{n=1}^N$. The classical binary arithmetic coding process for X uses the probabilities p_X and $(1-p_X)$ to partition the $[0,1)$ interval into sub-intervals associated to possible occurrences of the input symbols. At initialization the current interval is set to $I_0 = [0,1)$. For each input symbol, the current interval I_n is partitioned into two adjacent sub-intervals of lengths $(1-p_X)|I_n|$ and $p_X|I_n|$, where $|I_n|$ is the length of I_n. The sub-interval corresponding to the actual value of x_n is selected as the next current interval I_{n+1}, and this procedure is repeated for the next symbol. After all N symbols have been processed, the sequence is represented by the final interval I_N. The resulting codeword C_X can consist in the binary representation of any number inside I_N.

DAC is based on the principle of inserting some ambiguity in the source description during the encoding process. This is obtained employing a set of intervals whose lengths are proportional to the modified probabilities $\widetilde{p}_X^0 \geq (1-p_X)$ and $\widetilde{p}_X^1 \geq p_X$. In order to fit the enlarged sub-intervals into the $[0,1)$ interval, they are allowed to partially overlap. The encoding procedure is exactly the same as for arithmetic coding, but employs the modified probabilities. The codeword C_X is shorter, *i.e.*, the bit-rate is smaller, so much so as the interval overlap is large. The amount of overlap is a parameter that can be selected so as to achieve the desired rate R_X, which should be greater or equal to $H(X|Y)$.

In practice, the DAC encoding process has to be terminated properly in order to avoid clusters of errors at the end of the block. This can be done in several ways, e.g., encoding a known termination pattern or end-of-block symbol with a certain probability or, in the case of context-based AC, driving the AC encoder in a given context. For DAC, in this Section termination is obtained by encoding the last T symbols of the sequence without interval overlap. As T increases, the average error location tends to move toward the center of the block, yielding a correct decoder behavior. However, the termination has a cost in terms of bit-rate, as the last T symbols do not benefit from the Slepian-Wolf bit-rate saving.

DAC: decoding module

The DAC decoding process can be formulated as a symbol-driven sequential search along a proper decoding tree, where each node represents a state of the sequential arithmetic decoder. When the n-th input symbol x_n is decoded, if the codeword lies in overlapped region of the current interval then the decoder performs a branching. Two alternative paths are stored in the decoding memory, corresponding to the two alternative decoded symbols $x_n = 0$ and $x_n = 1$ that could be output at this step. For each new state, the associated branch metric is updated, and the corresponding interval is selected for next iteration. In particular, the correlated side information Y is employed to compute the *Maximum A Posteriori* (MAP) branch metric $\mathbb{P}(\mathbf{x}|C_X, \mathbf{y})$. In order to reduce complexity, after decoding a new input symbol,

the decoder keeps only the M paths with the best partial metric, and prunes the others; this is done using the M-algorithm. More details on the DAC encoding and decoding procedures can be found in [GMO09].

The DAC decoding algorithm is suboptimal, as the M-algorithm only keeps a limited number of likely decoding paths. If the correct path is dropped at some point during the decoding process, decoding will be unsuccessful. Thus, one would want to keep M very large in order to achieve the best performance, *i.e.*, the lowest residual error rate. On the other hand, M heavily affects the decoder complexity, as it directly impacts on the size of the search space.

4.1.1.4 Comparison of the Turbo and the arithmetic approaches

We carry out simulations based on the following conditions. We consider X with source distributions $p_X = \{0.5, 0.7\}$, of length $N = 1000$. For each source distribution, we consider the compression rates $1:2$ and $1:3$ for source X (R_X should be greater or equal to $H(X|Y)$ which depends on the correlation between the sources and on the source distribution); Y is compressed at its entropy $H(Y)$, which does not depend on the correlation between the sources. Actually, given the coding rate R_X, the source probability p_X and $H(X|Y)$, we compute the corresponding p s.t. $H(p_X) + H(p) - H(p_Y) = H(X|Y)$, where $p_Y = p_X(1-p) + (1-p_X)p$. In order to plot comparative curves, we consider different values of $H(X|Y)$ and measure the bit error rate (BER) between $\hat{\mathbf{x}}$ and $\mathbf{x}$. For each value of $H(X|Y)$, at least 10^7 bits are simulated. We choose the parameters of the DAC and the Turbo codes so as to obtain the same coding/decoding complexity.

For the system using Turbo codes, we utilize constituent Convolutional encoders defined by a constraint length $L = 4$ and by their octal parity-check matrix $\mathbf{H} = \begin{pmatrix} 11 & 15 & 06 \\ 15 & 12 & 17 \end{pmatrix}_{(oct)}$ of initial compression rate $2:3$, yielding an initial redundancy rate of $4:3$ for the resulting Turbo code. The different compression rates $1:2$ and $1:3$ are obtained with regular puncturing patterns of the syndrome. The interleaver is random and generated only once: the same interleaver is used for all the simulations. At most 20 decoding iterations are performed for each block of length N.

For the DAC, the desired compression rate is achieved by properly selecting the overlap factor, given the probability p_X, and taking into account the additional overhead generated by the termination. Throughout the simulations we employ $T = 20$. The rate achieved by the DAC over each data block is not always identical, but its deviation from the nominal value is very small. Over each simulation of 10^7 samples, the actual achieved rate is on average 0.2% smaller than the nominal rate. The value of M for the DAC decoder has been taken so that the complexity is roughly the same as the Turbo code system. To measure the complexities, both programs have been run on a Linux workstation, performing 200 rounds of encoding and decoding and measuring the running time. Taking $M = 1024$ yields similar times; this value has been used for all simulation results reported in this Section. While this procedure only yields a rough estimate of the complexity of the two systems, they are so different that an analytical complexity comparison is not viable.

Fig. 4.1 shows the BER achieved with the different methods, for different source probabilities and for different compression rates.

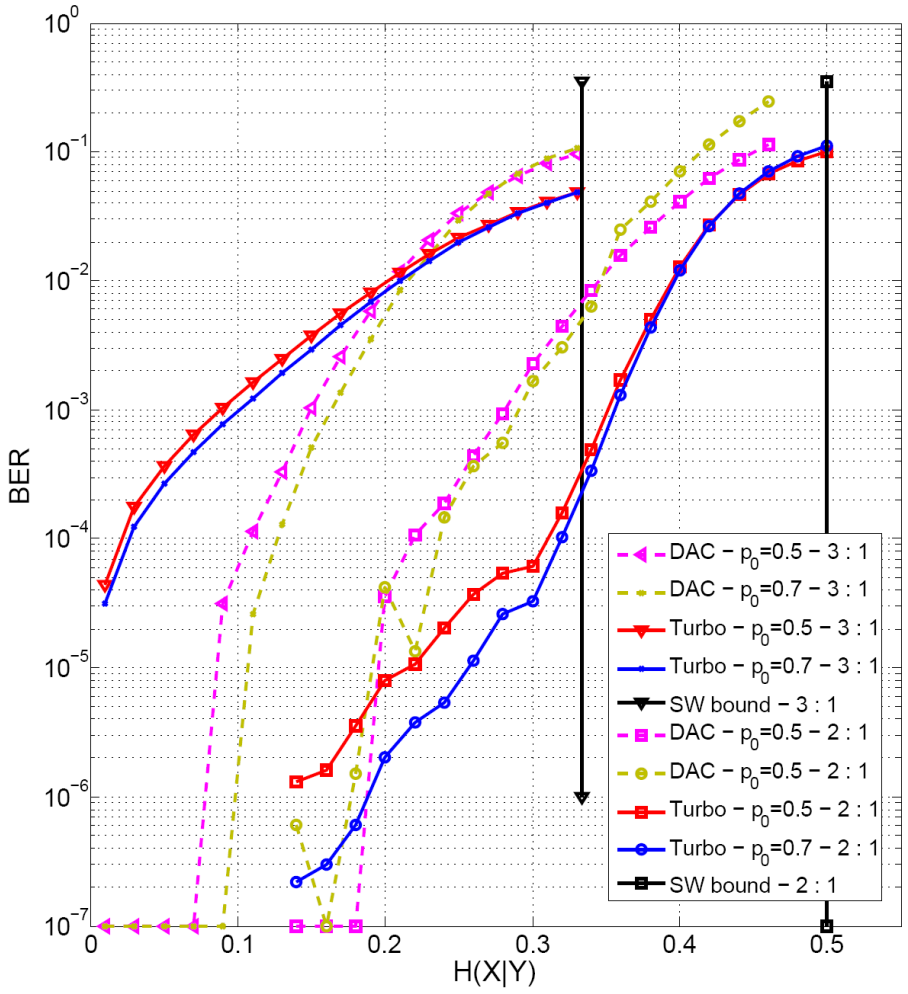

Figure 4.1: BER versus $H(X|Y)$ for DAC and Turbo codes. $N = 1000$, $p_X = 0.5$ and $p_X = 0.7$.

For compression rate $1 : 2$, the Turbo code is consistently better, except when the conditional entropy is very small ($H(X|Y) < 0.18$). This is because the DAC decoder is suboptimal, and this is particularly evident as the correlation decreases ($H(X|Y) > 0.2$). For compression rate $1 : 3$, the DAC is consistently better. The reason is that the heavier puncturing makes the syndrome based Turbo code less efficient. The Turbo code is still better at low correlation ($H(X|Y) > 0.22$), as the DAC suffers from the decoder sub optimality.

From $p_X = 0.5$ to $p_X = 0.7$, both the Turbo code and the DAC improve their distances to the Slepian-Wolf bounds. They clearly benefit from the prior knowledge of the source distribution. That gain should increase as the sources become less uniform, as expected from the theoretical rate gain shown in Fig. 3.1.

Finally, we have studied the DAC performance as the decoder complexity M varies. In particular, values of M ranging from 256 to 4096 have been taken. Simulations have been run for $p_X = 0.5$ and compression rate $1 : 2$. The results are shown in Fig. 4.2. As can be seen, between $M = 256$ and $M = 4096$ there is roughly an order of magnitude of BER difference. Interestingly, the performance gain does not tend to saturate in this range of M. This indicates that the sequential decoding

process is rather suboptimal, and the performance can be improved even more by further increasing M, although the computation times become prohibitive.

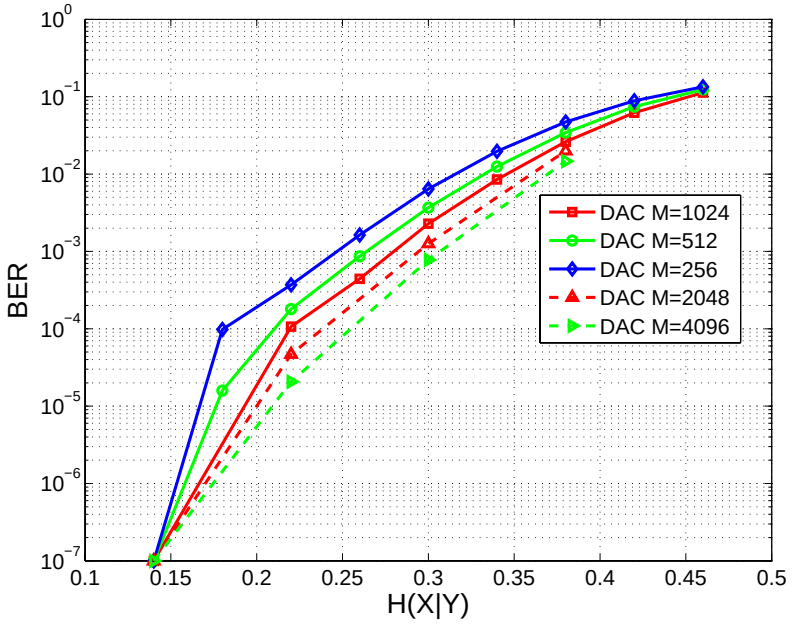

Figure 4.2: DAC performance as a function of M. $n = 1000$, $p_X = 0.5$.

4.1.2 Joint estimation-decoding for asymmetric coding of non-uniform sources using LDPC codes

We propose a modified LDPC decoder which accounts for the non-uniformity of the sources and which is adapted to the correlation model (additive or predictive) as well. The source distribution is iteratively estimated along with the symbol bits, and the reliability of this estimation is assessed. For an LDPC code yielding a compression rate $\frac{N-K}{N}$, we modify the standard decoding proposed by Liveris *et al.* ([LXG02] and Section 1.3.2.5) to exploit the non-uniformity of the sources and to deal with the type of BSC. The encoder is not modified.

4.1.2.1 LDPC decoding that accounts for the source distribution

Consider the following notation and definition of the messages that are passed in the graph.

- $x_n, n \in [1, N]$ are the source symbols, represented by the *variable nodes*; their estimates are noted $\hat{x}_n$
- $y_n, n \in [1, N]$ are the side-information symbols, represented by the *side-information nodes*;
- $s_m, m \in [1, (N-K)]$ are the syndrome symbols, represented by the check nodes;
- d_{xn} is the *degree* of x_n;

- d_{sm} is the *degree* of s_m;
- $I_n, n \in [1, N]$ are the *intrinsic*, passed from y_n to x_n;
- $E_{n,e}, n \in [1, N], e \in [1, d_{xn}]$ are the extrinsic information, passed from x_n on their e-th edge to the check nodes;
- $Q_{m,e}, m \in [1, (N-K)], e \in [1, d_{sm}]$ are the messages passed from s_m on their e-th edge to the variable nodes;
- $\hat{p}_X$ denotes the estimate of p_X. It is updated throughout the iterations, after each update of $\hat{\mathbf{x}}$.

All the messages are Log-Likelihood Ratio (LLR), they are labeled (*in*) or (*out*) if they come *to* or *from* the nodes.

Intrinsic information computation The intrinsic information depends on the type of BSC. It is defined by:

$$I_n(p_X) = \log\left(\frac{\mathbb{P}(X_n = 0|y_n)}{\mathbb{P}(X_n = 1|y_n)}\right) = \begin{cases} (1-2y_n)\log\left(\dfrac{1-p}{p}\right), \ if\ the\ BSC\ is\ predictive \\ (1-2y_n)\log\left(\dfrac{1-p}{p}\right) + \log\left(\dfrac{1-p_X}{p_X}\right), \ if\ additive \end{cases} \tag{4.2}$$

Since p_X is *not* known *a priori*, each I_n is initialized to $I_n(\hat{p}_Y)$ where $\hat{p}_Y$ is the probability of 1's in Y, that is the best guess on p_X so far. Each $E_{n,k}^{(in)}$ is initialized to 0. Note that the initialization of p_X can also be done using the Bernoulli parameter estimator presented in Section 3.4, using the syndrome $\mathbf{s_x}$, provided that $H(p_X)$ is lower than $\frac{N-K}{N}$.

Messages from the variable nodes to the check nodes

$$E_{n,e}^{(out)} = I_n(\hat{p}_X) + \sum_{k=1, k\neq e}^{d_{xn}} E_{n,k}^{(in)}$$

where $I_n(\cdot)$ is defined in Equation (4.2). Each extrinsic message $E_{n,e}^{(out)}$ is mapped to the corresponding $Q_{m,e}^{(in)}$ according to the connections in the graph.

Messages from the check nodes to the variable nodes

$$Q_{m,e}^{(out)} = 2\tanh^{-1}\left[(1-2s_n)\prod_{k=1, k\neq e}^{d_{sm}} \tanh\frac{Q_{m,e}^{(in)}}{2}\right]$$

Each $Q_{m,e}^{(out)}$ is mapped to the corresponding $E_{n,e}^{(in)}$.

Decision, and update of $\hat{p}_X$

We denote $E_n = I_n(\hat{p}_X) + \sum_{k=1}^{d_{xn}} E_{n,k}^{(in)} = \log\left(\frac{1-\mathbb{P}_n}{\mathbb{P}_n}\right)$
where $\mathbb{P}_n$ is the best guess on $\mathbb{P}(X_n = 1|\mathbf{y}, \mathbf{s_x})$ so far. Then

$$\mathbb{P}_n = \frac{e^{E_n}}{1 - e^{E_n}},\ and\ \hat{x}_n = \begin{cases} 1,\ if\ \mathbb{P}_n \geq 0.5 \\ 0,\ otherwise \end{cases}$$

$\hat{p}_X$ is estimated as the probability of 1's in $\hat{\mathbf{x}}$, expressed with the soft values $\mathbb{P}_n$. The updated value of $\hat{p}_X$ is thus given by

$$\hat{p}_X = \frac{\sum_{n=1}^{N} \mathbb{P}_n}{N} \tag{4.3}$$

This update rule corresponds to that obtain by an EM algorithm.

Stopping criteria: syndromes check, convergence test, and maximum number of iterations

The decoding algorithm stops either if the estimated $\hat{\mathbf{x}}$ satisfies the parity check equation ($\mathbf{H}\hat{\mathbf{x}} = \mathbf{s_x}$), or if the maximal number of iterations has been reached (100 iterations is a good compromise between performance and complexity). Moreover if the syndrome test has failed, while no symbols of $\hat{\mathbf{x}}$ have been updated during the current iteration, even if the maximal number of iterations has not been reached yet, we decide that the decoder has converged to a wrong word.

4.1.2.2 Simulation results

Non-uniformity and additive BSC

We test the proposed decoding algorithm using an LDPC code of rate $\frac{1}{2}$ created using the Progressive Edge Growth (PEG) principle ([HEA05] and Annex B). The non-uniform sources are drawn for two values of $p_X = \{0.15, 0.2275\}$; and, for each p_X, a range of values of p is considered. The source sequences are of length $N = 1584$ (which corresponds to the bit planes block length in the DVC experiments). The syndrome $\mathbf{s_x}$, as well as the side-information $\mathbf{y}$, are transmitted to *three* different decoders:

- ① the standard decoder that views X as a uniform source;
- ② the proposed decoder that knows that X is non-uniform and has to estimate $\hat{p}_X$;
- ③ a *genie-aided* decoder that knows p_X (in order to quantify the sub-optimality introduced by the parallel estimation of $\hat{p}_X$).

When the BSC is additive, Fig. 4.3 shows the performance of the decoding in terms of its Bit Error Rate (BER) versus $H(p)$.

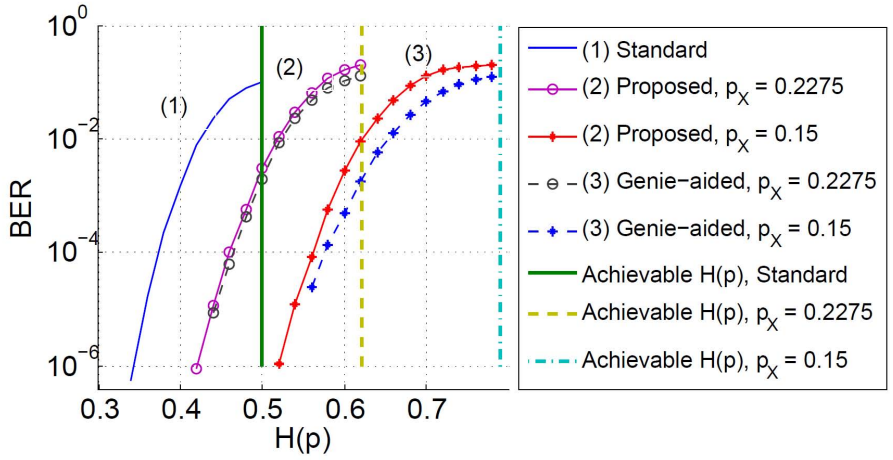

Figure 4.3: Performance of the standard and the modified LDPC decoders, for non-uniform sources with parameters $p_X = \{0.15, 0.2275\}$, over an additive BSC.

The standard decoder ① has the same performance regardless of the source distribution. Meanwhile, the decoders ② and ③ exploiting the non-uniformity are able to retrieve X from considerably greater $H(p)$; the rate gain increases with the non-uniformity. When considering that X is uniform, decoder ①, $H(X|Y) = 0.5$ is achieved for $H(p) = 0.5$ ($p = 0.11$) regardless of the source distribution. When exploiting the non-uniformity, decoders ② and ③, $H(X|Y) = 0.5$ occurs for $H(p) = 0.622$ ($p = 0.155$) when $p_X = 0.2275$, and for $H(p) = 0.79$ ($p = 0.239$) when $p_X = 0.15$. The estimation of the source Bernoulli parameter, only induces a loss lower than $0.02bit$ when the BER is under 10^{-5}, which is acceptable regarding the rate gain with respect to the standard decoder.

Effect of a mismatch between the true correlation model and the one assumed by the decoder

Now, we assess the impact of a wrong guess of the type of BSC between the correlated sources. Let us first consider the case where the true correlation model is *additive* while the decoder assumes a *predictive* one. The curve ① in Fig. 4.3 shows the performance of such a mismatched decoder while curves ② and ③ show the performance of the matched decoder. The mismatch induces a significant rate loss. For the other case, we consider a *predictive* correlation channel model, while the decoder is configured to take into account an *additive* one. We plot the BER of that system on Fig. 4.4, along with that of the standard decoder.

The mismatched decoder performs worse than the correct decoder. This result might seem counter-intuitive since exploiting the source non-uniformity "should" always improve the performance of the decoder. However, consider that the best decoder that can be implemented is the matched decoder, so a mismatched decoder always performs worse; in that sense, this result might seem obvious.

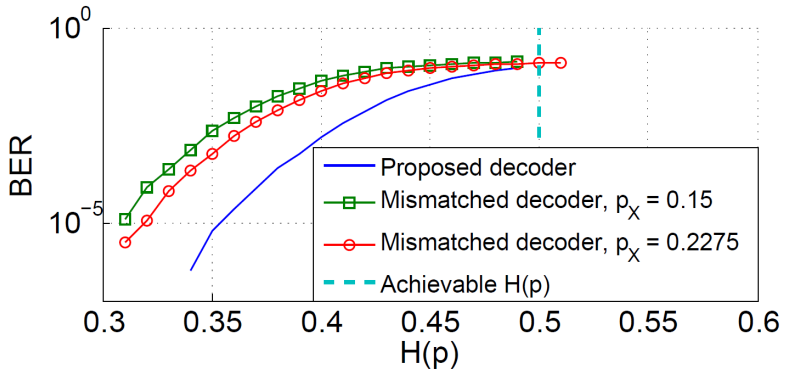

Figure 4.4: Influence of a mismatch on the channel type (the BSC is assumed to be additive while it is predictive), for two non-uniform sources having parameters $p_X = \{0.15, 0.2275\}$.

Performance for the parallel parameter estimation

Fig. 4.5 shows the estimated parameters, averaged over $5 \cdot 10^3$ realizations of the source.

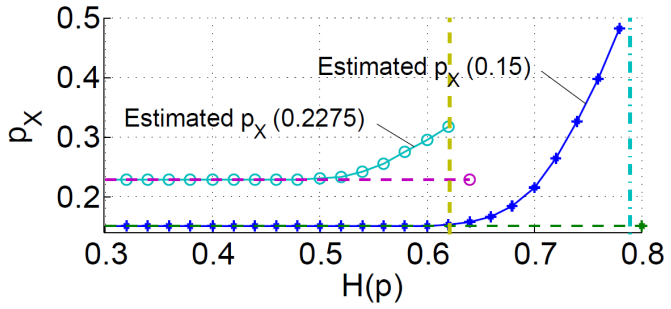

Figure 4.5: Performance of the parameter estimator that is implemented, for non-uniform sources of parameters $p_X = \{0.15, 0.2275\}$.

When the correlation level is such that $H(p)$ is far from the SW bound ($H(p) < 0.52$ for $p_X = 0.2275$, and $H(p) < 0.62$ for $p_X = 0.15$), the decoding of X is successful (BER$< 10^{-2}$) and the parameter is well estimated (the bias of the estimator is lower than 10^{-3}); but when the correlation is lower, the decoding of X fails and the parameter estimation fails as well.

4.2 Asymmetric coding of Gilbert-Elliott sources

In Section 3.2, we presented the GE source model, the achievable rate bounds that can be reached for the SW coding, when the BSC modeling the correlation is *additive* or *predictive*, and we presented a particular implementation of the EM algorithm for the estimation of θ_X when the source realization $\mathbf{x}$ is available (the Baum-Welch algorithm, in Section 3.2.4). We also showed that the GE model is a better model than

non-uniform sources, for the bit planes generated by the DVC codec DISCOVER in Section 5.2.1. This Section is dedicated to the task of estimating the source parameters when only corrupted observations of $\mathbf{x}$ are available, namely its syndrome $\mathbf{s_x}$ and the side-information $\mathbf{y}$. A Turbo decoder that takes into account the statistics of the GE channel is presented in [GFV02] using super-trellises, the usual decoders in the literature place a channel interleaver and do the decoding assuming that the channel is memoryless. Later, Eckford investigated the use of LDPC codes for decoding of Bernoulli sources over the GE channel [Eck04, Eck05].

4.2.1 Formal statement of the problem and the EM algorithm

Here, the SW decoder must estimate the source realization $\mathbf{x}$ from its syndrome $\mathbf{s_x}$ and the side-information $\mathbf{y}$, knowing *a priori* that X is a GE source. However, the decoder is not aware neither of the parameter θ_X nor of the hidden state Σ. This estimation is carried out with an *Expectation-Maximization* (EM) algorithm, that learns new parameters from the observed variables, $\mathbf{y}$ and $\mathbf{s_x}$, and from a previous estimation of the hidden variables, $\hat{\mathbf{x}}$, $\hat{\sigma}_{\mathbf{x}}$ and $\hat{\theta}_X$.

Let l be the label of the current iteration of the EM, and θ_X^l the current estimate of the GE source parameters. Then, the updated value $\theta_X^{(l+1)}$ corresponding to the next iteration is computed so as to maximize the following mean log-likelihood function:

$$\theta_X^{(l+1)} = \arg\max_{\theta_X} \left(\mathbb{E}_{\mathbf{X},\mathbf{\Sigma_x}|\mathbf{Y},\mathbf{S_X},\theta_X^l} \left[\log \left(\mathbb{P}_{\theta_X}(\mathbf{y}, \mathbf{x}, \sigma_{\mathbf{x}}, \mathbf{s_x}) \right) \right] \right) \tag{4.4}$$

To simplify the notation, in the sequel we denote "$\mathbb{P}(\mathbf{X} = \mathbf{x}|\theta_X)$" by "$\mathbb{P}_{\theta_X}(\mathbf{x})$".

Since the logarithm is a strictly increasing function, the value $\theta_X^{(l+1)}$ that maximizes $\mathbb{P}_{\theta_X^l}(\mathbf{y}, \mathbf{x}, \sigma_{\mathbf{x}}, \mathbf{s_x})$ also maximizes $\log\left(\mathbb{P}_{\theta_X^l}(\mathbf{y}, \mathbf{x}, \sigma_{\mathbf{x}}, \mathbf{s_x})\right)$. The algorithm converges since it increases the likelihood at each iteration [Wu83].

We now consider that the code used to compress the source X is a syndrome-based LDPC code. For an LDPC code yielding a compression rate $\frac{N-K}{N}$, let $\mathbf{H} = (h_{mn})_{m\in[1,(N-K)],n\in[1,N]}$ be the sparse matrix of size $(N-K)\times N$ corresponding to the relationships between the variables. Fig. 4.6 presents the factor graph [KFL01] that describes the dependencies between the observed and the hidden variables of the problem.

We introduce the following notation for variables and messages that are passed on the factor graph during the estimation-decoding process:

- x_n are the source symbols, represented by the *variable nodes*; their estimates are denoted by $\hat{x}_n$;
- y_n are the side-information symbols, represented by the *side-information nodes*;
- z_n are the noise realization symbols, represented by the *BSC nodes*;
- s_m are the syndrome symbols, represented by the *check nodes*;
- d_{xn} is the *degree* of x_n, *i.e.* the number of check nodes connected to it;
- d_{sm} is the *degree* of s_m, *i.e.* the number of variable nodes connected to it;

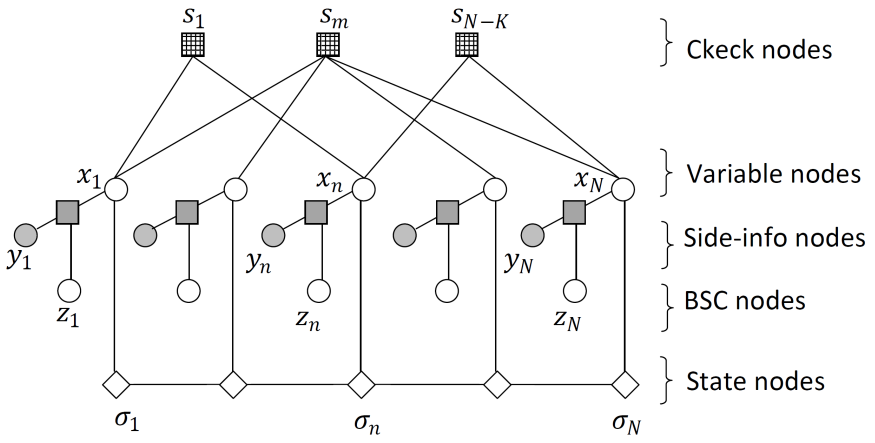

Figure 4.6: Factor graph describing the joint estimation-decoding EM.

- I_n is the intrinsic information for the node z_n;
- $E_{n,e}, e \in [1, d_{xn}]$ are the messages passed from the variable nodes, on their e-th edge, to the check nodes;
- E_n is the *a posteriori* LLR of $\hat{x}_n$;
- $Q_{m,e}, e \in [1, d_{sm}]$ are the messages passed from the check nodes, on their e-th edge, to the variable nodes;
- B_n are the messages passed from the BSC node z_n to the variable node x_n;
- S_n : Messages from the state node σ_n to the variable node x_n;
- V_n : Messages from the variable node x_n to the state node σ_n.

All the messages are Log-Likelihood Ratio (LLR), they are labeled (*in*) or (*out*) if they come *to* or *from* the considered node. In the following, we give the update rules for the messages that are passed.

4.2.2 Expectation step: computation of the mean log-likelihood function

We expand the likelihood function using the Bayes rule:

$$\mathbb{P}_{\theta_X^l}(\mathbf{y}, \mathbf{x}, \sigma_\mathbf{x}, \mathbf{s_x}) = \mathbb{P}(\mathbf{y}, \mathbf{s_x}|\mathbf{x}, \sigma_\mathbf{x})\mathbb{P}_{\theta_X^l}(\mathbf{x}, \sigma_\mathbf{x})$$

where $\mathbb{P}(\mathbf{y}, \mathbf{s_x}|\mathbf{x}, \sigma_\mathbf{x})$ is independent of θ_X^l.

The log-likelihood function is thus expressed as:

$$\begin{aligned}\log\left(\mathbb{P}_{\theta_X^l}(\mathbf{x}, \sigma_\mathbf{x})\right) = \log\left(\mathbb{P}_{\theta_X^l}(\sigma_1)\right) + \sum_{n=2}^{N}\sum_{i=\mathbf{0}}^{\mathbf{1}}\sum_{j=\mathbf{0}}^{\mathbf{1}} \delta_{\sigma_{n-1}=i,\sigma_n=j}\log(t_{ij}^l) \\ + \sum_{n=1}^{N}\sum_{i=\mathbf{0}}^{\mathbf{1}} \delta_{\sigma_n=i,x_n=1}\log(p_i^l) + \delta_{\sigma_n=i,x_n=0}\log(1-p_i^l)\end{aligned} \tag{4.5}$$

where $\delta_{bool} = \begin{cases} 1, \ if \ bool = true \\ 0, \ otherwise \end{cases}$ is the Kroenecker's symbol.

Finally, the mean log-likelihood function is obtained by taking the expectation of the log-likelihood function (4.5), where

$$\begin{aligned} \mathbb{E}_{\mathbf{X},\mathbf{\Sigma_x}|\mathbf{Y},\mathbf{S_X},\theta_X^l}\left[\delta_{\sigma_n=i,x_n=k}\right] &= \mathbb{P}_{\theta_X^l}(\Sigma_n = i, X_n = k|\mathbf{y},\mathbf{s_x}) \\ &= \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{y},\mathbf{s_x})\mathbb{P}_{\theta_X^l}(X_n = k|\Sigma_n = i,\mathbf{y},\mathbf{s_x}) \quad (4.6) \\ \mathbb{E}_{\mathbf{X},\mathbf{\Sigma_x}|\mathbf{Y},\mathbf{S_X},\theta_X^l}\left[\delta_{\sigma_{n-1}=i,\sigma_n=j}\right] &= \mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j|\mathbf{y},\mathbf{s_x}) \end{aligned}$$

From Equations (4.5) and (4.6), the mean log-likelihood function is thus given by:

$$\begin{aligned} &\mathbb{E}_{\mathbf{X},\mathbf{\Sigma_x}|\mathbf{Y},\mathbf{S_X},\theta_X^l}\left[\log\left(\mathbb{P}_{\theta_X^l}(\mathbf{x},\sigma_\mathbf{x})\right)\right] = \log\left(\mathbb{P}_{\theta_X^l}(\sigma_1)\right) \\ &+ \sum_{n=2}^{N}\sum_{i=\mathbf{0}}^{\mathbf{1}}\sum_{j=\mathbf{0}}^{\mathbf{1}}\mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j|\mathbf{y},\mathbf{s_x})\log(t_{ij}^l) \\ &+ \sum_{n=1}^{N}\sum_{i=\mathbf{0}}^{\mathbf{1}}\mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{y},\mathbf{s_x})\mathbb{P}_{\theta_X^l}(X_n = 1|\mathbf{y},\mathbf{s_x})\log(p_i^l) \\ &+ \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{y},\mathbf{s_x})\mathbb{P}_{\theta_X^l}(X_n = 0|\mathbf{y},\mathbf{s_x})\log(1-p_i^l) \end{aligned} \quad (4.7)$$

Note that this expression of $\mathbb{E}_{\mathbf{X},\mathbf{\Sigma_x}|\mathbf{Y},\mathbf{S_X},\theta_X^l}\left[\log\left(\mathbb{P}_{\theta_X^l}(\mathbf{x},\sigma_\mathbf{x})\right)\right]$ is derivable in θ_X.

4.2.3 Maximization step: update rules for the parameters

Here, the mean log-likelihood function is maximized versus θ_X, given the estimate θ_X^l, and under the three constraints, $\forall i,j \in \{\mathbf{0},\mathbf{1}\}$:

$$p_i \in [0,1], \ and \ t_{ij} \in \ ]\,0,1\,[\ , \ and \sum_{k\in\{\mathbf{0},\mathbf{1}\}} t_{ik} = 1 \quad (4.8)$$

The partial derivatives of the mean log-likelihood function (4.7), with respect to the Bernoulli parameters are, $\forall i \in \{\mathbf{0},\mathbf{1}\}$:

$$\begin{aligned} &\frac{\partial}{\partial p_i^l}\mathbb{E}_{\mathbf{X},\mathbf{\Sigma_x}|\mathbf{Y},\mathbf{S_X},\theta_X^l}\left[\log\left(\mathbb{P}_{\theta_X^l}(\mathbf{x},\sigma_\mathbf{x})\right)\right] = \\ &\left(\frac{1}{p_i^l}\right)\sum_{n=1}^{N}\mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{y},\mathbf{s_x})\mathbb{P}_{\theta_X^l}(X_n = 1|\mathbf{y},\mathbf{s_x}) + \\ &\left(\frac{1}{1-p_i^l}\right)\sum_{n=1}^{N}\mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{y},\mathbf{s_x})\mathbb{P}_{\theta_X^l}(X_n = 0|\mathbf{y},\mathbf{s_x}) \end{aligned} \quad (4.9)$$

Then, the derivative (4.9) is zero when, $\forall i \in \{\mathbf{0},\mathbf{1}\}$:

$$p_i^{(l+1)} = \frac{\sum_{n=1}^{N} \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{y}, \mathbf{s_x}) \mathbb{P}_{\theta_X^l}(X_n = 1|\Sigma_n = i, \mathbf{y}, \mathbf{s_x})}{\sum_{n=1}^{N} \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{y}, \mathbf{s_x})} \tag{4.10}$$

For the transition parameters, given that $t_{\mathbf{00}} = (1 - t_{\mathbf{01}})$ and $t_{\mathbf{11}} = (1 - t_{\mathbf{10}})$ (3.3), the partial derivatives are given by, $\forall i, j \in \{\mathbf{0}, \mathbf{1}\}, i \neq j$:

$$\begin{aligned} &\frac{\partial}{\partial t_{ij}^l} \mathbb{E}_{\mathbf{X}, \mathbf{\Sigma_x}|\mathbf{Y}, \mathbf{S_X}, \theta_X^l} \left[\log \left(\mathbb{P}_{\theta_X^l}(\mathbf{x}, \sigma_\mathbf{x}) \right) \right] = \\ &\left(\frac{1}{t_{ij}^l} \right) \sum_{n=2}^{N} \mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j|\mathbf{y}, \mathbf{s_x}) \mathbb{P}_{\theta_X^l}(X_n = 1|\mathbf{y}, \mathbf{s_x}) + \\ &\left(\frac{1}{1 - t_{ij}^l} \right) \sum_{n=2}^{N} \mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = i|\mathbf{y}, \mathbf{s_x}) \mathbb{P}_{\theta_X^l}(X_n = 0|\mathbf{y}, \mathbf{s_x}) \end{aligned} \tag{4.11}$$

Then, the derivative (4.11) is zero when, $\forall i, j \in \{\mathbf{0}, \mathbf{1}\}, i \neq j$:

$$t_{ij}^{(l+1)} = \frac{\sum_{n=2}^{N} \mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j, \mathbf{y}, \mathbf{s_x})}{\sum_{n=1}^{N-1} \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{y}, \mathbf{s_x})} \tag{4.12}$$

Note that the constraints in Equation (4.8) are met by the solutions that are exhibited, provided that the parameters are correctly normalized.

Therefore, the maximization step needs the current *a posteriori probabilities* (APP) of the states Σ and the source X. Due to the cycles in the graph, these quantities are too complex to compute, but they can efficiently be approximated by a *belief propagation* algorithm run on the graph shown in Fig. 4.6. The algorithm proceeds in two steps explained below: the soft estimates of the states result from a *forward-backward-like algorithm*, and the soft estimates of the source symbols result from an *LDPC-decoding-like* algorithm.

4.2.4 Forward-backward algorithm for soft estimates of the states

Here, the probabilities of $\mathbf{\Sigma}$ are updated according to the values of the estimate θ_X^l. The aim is to compute

$$\begin{aligned} \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{y}, \mathbf{s_x}) &= \frac{\mathbb{P}_{\theta_X^l}(\Sigma_n = i, \mathbf{y}|\mathbf{s_x})}{\mathbb{P}_{\theta_X^l}(\mathbf{y}|\mathbf{s_x})} \\ \mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j|\mathbf{y}, \mathbf{s_x}) &= \frac{\mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j, \mathbf{y}|\mathbf{s_x})}{\mathbb{P}_{\theta_X^l}(\mathbf{y}|\mathbf{s_x})} \end{aligned}$$

To that end, we decompose the following expression, to retrieve the equations corresponding to the forward-backward recursions:

$$\begin{aligned}\mathbb{P}_{\theta_X^l}(\Sigma_n = i, \mathbf{y}|\mathbf{s_x}) &= \sum_{j\in\{\mathbf{0},\mathbf{1}\}} \mathbb{P}_{\theta_X^l}(\Sigma_n = i, \Sigma_{n+1} = j, \mathbf{y}|\mathbf{s_x}) \\ &= \sum_{j\in\{\mathbf{0},\mathbf{1}\}} \alpha_i^n \cdot \gamma_{i,j}^{n,(n+1)} \cdot \beta_j^{(n+1)}\end{aligned}$$

The forward-backward algorithm is run on the trellis in Fig. 4.7 defined by the states **0** and **1** generating the source symbols, with two branches between the states, labeled by the two possible values $x_n = 0$ and $x_n = 1$.

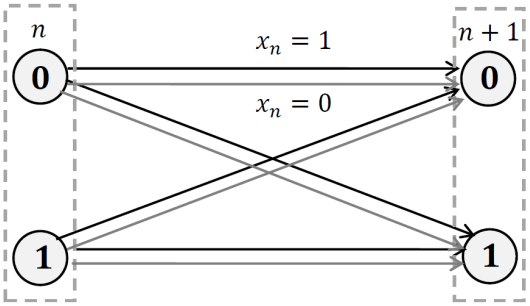

Figure 4.7: Trellis on which the forward-backward algorithm is run to estimate the underlying states $\mathbf{\Sigma}$ of the GE process.

We define

$$\begin{aligned}\gamma_{i,j}^{n,(n+1)} =& \mathbb{P}_{\theta_X^l}(y_n|\Sigma_n = i, \mathbf{s_x}) \cdot \mathbb{P}_{\theta_X^l}(\Sigma_{n+1} = j|\Sigma_n = i, \mathbf{s_x}) \\ \alpha_j^n =& \sum_{i\in\{\mathbf{0},\mathbf{1}\}} \alpha_i^{(n-1)} \cdot \gamma_{i,j}^{(n-1),n} \\ \beta_i^n =& \sum_{i\in\{\mathbf{0},\mathbf{1}\}} \gamma_{i,j}^{n,(n+1)} \cdot \beta_j^{(n+1)}\end{aligned}$$

where:

- $\gamma_{i,j}^{n,(n+1)}$ is the transition probability between the states i at position n and j at position $(n+1)$.
- α_j^n is the forward probability for the source to be in state j at position n;
- β_i^n is the backward probability for the source to be in state i at position n.

Now we define the states APP:

$$\begin{aligned}\mathbb{P}_{\theta_X^l}(\Sigma_n = i, \mathbf{y}|\mathbf{s_x}) &= \alpha_i^n \cdot \beta_i^n \\ \mathbb{P}_{\theta_X^l}(\Sigma_{n-1} = i, \Sigma_n = j, \mathbf{y}|\mathbf{s_x}) &= \alpha_i^{(n-1)} \cdot \gamma_{i,j}^{(n-1),n} \cdot \beta_j^n\end{aligned}$$

Normalizing $\mathbb{P}_{\theta_X^l}(\sigma_n, \mathbf{y}|\mathbf{s_x})$ and $\mathbb{P}_{\theta_X^l}(\sigma_{n-1}, \sigma_n, \mathbf{y}|\mathbf{s_x})$, we get the desired values $\mathbb{P}_{\theta_X^l}(\sigma_n|\mathbf{y}, \mathbf{s_x})$ and $\mathbb{P}_{\theta_X^l}(\sigma_{n-1}, \sigma_n|\mathbf{y}, \mathbf{s_x})$.

4.2.5 LDPC decoding for the soft estimate of the source

Given the current estimate θ_X^l and the soft estimate $\sigma_{\mathbf{x}}^l$, we find the best approximation of the *a posteriori* probabilities (APP) $\mathbb{P}_{\theta_X^l}(X_n = k|\sigma_n, \mathbf{y}, \mathbf{s_x}), n \in [1, N], k \in \{0, 1\}$ needed for the parameters updates in Equations (4.10) and (4.12) of the maximization step. As side products, we also obtain the l-th estimate $\mathbf{x}^l$. Here, we describe the update rules for the belief-propagation run on the graph in Fig. 4.6.

Messages from the state nodes to the variable nodes

$$S_n = \log\left(\frac{\mathbb{P}_{\theta_X^l}(X_n = 0)}{\mathbb{P}_{\theta_X^l}(X_n = 1)}\right) = \log\left(\frac{1 - p_{Xn}^l}{p_{Xn}^l}\right)$$

where p_{Xn}^l is obtained from the states probabilities $\mathbb{P}_{\theta_X^l}(\sigma_n|\mathbf{y}, \mathbf{s_x})$ (from the forward-backward algorithm), and the current estimate θ_X^l. More precisely, $\forall n \in [1, N]$:

$$p_{Xn}^l = \sum_{i \in [\mathbf{0},\mathbf{1}]} p_i^l \cdot \mathbb{P}_{\theta_X^l}(\Sigma_n = i|\mathbf{y}, \mathbf{s_x}) \tag{4.13}$$

Intrinsic information of the BSC nodes

$$I_n = \log\left(\frac{\mathbb{P}(Z_n = 0)}{\mathbb{P}(Z_n = 1)}\right) = \log\left(\frac{1 - p}{p}\right)$$

Messages from the BSC nodes to the variable nodes

$$B_n = (1 - 2y_n)I_n \tag{4.14}$$

Messages from the variable nodes to the check nodes

$$E_{n,e}^{(out)} = \begin{cases} B_n + \sum_{k=1,k\neq e}^{d_{xn}} E_{n,k}^{(in)}, \ if \ predictive \ BSC \\ B_n + \sum_{k=1,k\neq e}^{d_{xn}} E_{n,k}^{(in)} + S_n, \ if \ additive \ BSC \end{cases}$$

Each $E_{n,e}^{(out)}$ is mapped to the corresponding $Q_{m,e}^{(in)}$ according to the connections in the factor graph.

Messages from the check nodes to the variable nodes

$$Q_{m,e}^{(out)} = 2\tanh^{-1}\left[(1 - 2s_n)\prod_{k=1,k\neq e}^{d_{sm}} \tanh\frac{Q_{m,e}^{(in)}}{2}\right]$$

Each $Q_{m,e}^{(out)}$ is mapped to the corresponding $E_{n,k}^{(in)}$.

Messages from the variable nodes to the state nodes

We compute the *extrinsic* LLR E_n for each x_n, as:

$$E_n = B_n + \sum_{k=1}^{d_{xn}} E_{n,k}^{(in)}$$

Then, the variable to state messages are given by:

$$\begin{cases} V_n(0) = \dfrac{e^{E_n}}{1+e^{E_n}} \\ V_n(1) = 1 - V_n(0) \end{cases} \tag{4.15}$$

For this LDPC decoding, we have decided to propagate LLR, which implies their conversion to probabilities, in (4.15), for use in the maximization step. So far, the values of $V_n(0)$ and $V_n(1)$ are the best guess on the *a posteriori* probabilities $\mathbb{P}_{\theta_X^l}(X_n = 0|\sigma_n, \mathbf{y}, \mathbf{s_x})$ and $\mathbb{P}_{\theta_X^l}(X_n = 1|\sigma_n, \mathbf{y}, \mathbf{s_x})$.

Decision After each iteration of the EM algorithm, a hard decision is made on $V_n(1)$ to get the estimated symbols of $\mathbf{x}^{(l+1)}$.

$$\forall n \in [1, N], x_n^{(l+1)} = \begin{cases} 1, \ if \ V_n(1) \geq 0.5 \\ 0, \ otherwise \end{cases}$$

4.2.6 Stopping criteria: syndrome check, convergence test, and maximum number of iterations

The EM estimation-decoding algorithm stops either if the estimated $\mathbf{x}^{(l+1)}$ satisfies the parity check equations defined by $\left(\mathbf{H}\mathbf{x}^{(l+1)} = \mathbf{s_x}\right)$, or if the syndrome test has failed while no symbol of $\hat{\mathbf{x}}$ has been updated during the current iteration (we decide that the decoder has converged to a wrong word), or if the maximum number of iterations has been reached (100 iterations is a good compromise between performance and complexity).

4.2.7 Initialization of the EM algorithm

As exhibited in Lemma 3.1, the side-information Y is a GE source that has the same states as the source X. Therefore, to initialize the EM algorithm, we use the estimated GE parameters associated to $\mathbf{y}$, $\{\hat{\theta}_Y, \hat{\sigma}_\mathbf{y}\}$; that is the best guess on $\{\theta_X^0, \sigma_\mathbf{x}^0\}$ so far. To estimate these parameters, we use the Baum-Welch algorithm presented in Section 3.2.4. This simplified EM algorithm is initialized with the arbitrary values $(p_\mathbf{0}^0 = 0.49, p_\mathbf{1}^0 = 0.51, t_\mathbf{10}^0 = 0.1, t_\mathbf{01}^0 = 0.1)$.

4.2.8 Simulation results for Distributed coding of GE sources

We consider a GE source X with parameters $\theta_X = (p_\mathbf{0} = 0.07, p_\mathbf{1} = 0.7, t_\mathbf{10} = 0.03, t_\mathbf{01} = 0.01)$, having realization $\mathbf{x}$ of length $N = 1584$ (same length as the video

bit planes, in the DVC experiments with QCIF sequences reported in Chapter 5). We also consider a noise $Z \sim \mathcal{B}(p)$ with realization $\mathbf{z}$, and a second source Y, with realization $\mathbf{y}$, correlated to X s.t. $Y = X \oplus Z$ (the BSC is *additive*). To prove the enhanced performance of the proposed DSC decoder, the syndrome of $\mathbf{x}$, as well as the side-information $\mathbf{y}$, are transmitted to *three* different decoders:

- ⓐ the *standard* decoder, which views X as a uniform source;
- ⓑ the *proposed* decoder, which knows that X is a GE source and uses the EM algorithm to iteratively estimate its parameters θ_X;
- ⓒ a *genie-aided* decoder, which knows the parameters θ_X.

The BER of X corresponding to the *three* decoders are plotted in Fig 4.8, the estimated parameters from the decoder ⓑ are plotted in Fig. 4.9. When exploiting the memory, $H(\mathcal{X}|\mathcal{Y}) = 0.5$ occurs for $H(p) = 0.88$ ($p = 0.299$), instead of $H(p) = 0.5$ ($p = 0.11$) when considering the source as uniform.

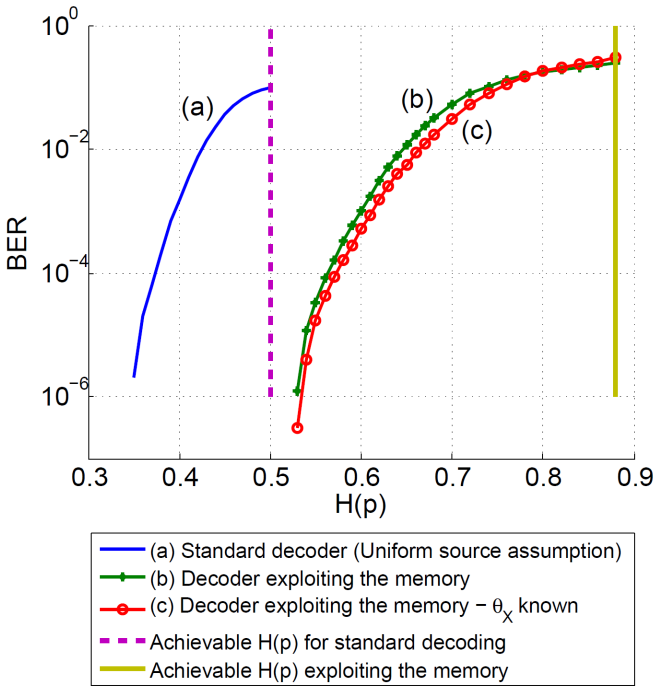

Figure 4.8: Performances of the *three* decoders, for a GE source X of parameter $\theta_X = (p_0 = 0.07, p_1 = 0.7, t_{10} = 0.03, t_{01} = 0.01)$.

First, we see in Fig. 4.8 that the proposed decoder ⓑ performs considerably better than the original decoder ⓐ. In particular, the source is retrieved error-free for $H(p) = 0.5$, which corresponds to the SW bound for uniform sources. Moreover, the plots ⓑ and ⓒ in Fig. 4.8 show that knowing the true θ_X is not essential to the decoder, since it does not improve the performance of the decoder in a significant amount: the rate improvement from ⓑ to ⓒ is less than 0.02bit for any value taken by p. The decoder presented in [GFV97] is not able to reach this bound since no estimation of the states is performed, their decoding is done on the super trellis of a

modified Turbo code, with a number of states that is equal to the number of states of the code *times* the number of states in the memory source. If a Turbo code was used for our scheme, the number of states would be the number of states of the code *plus* the number of states in the memory source.

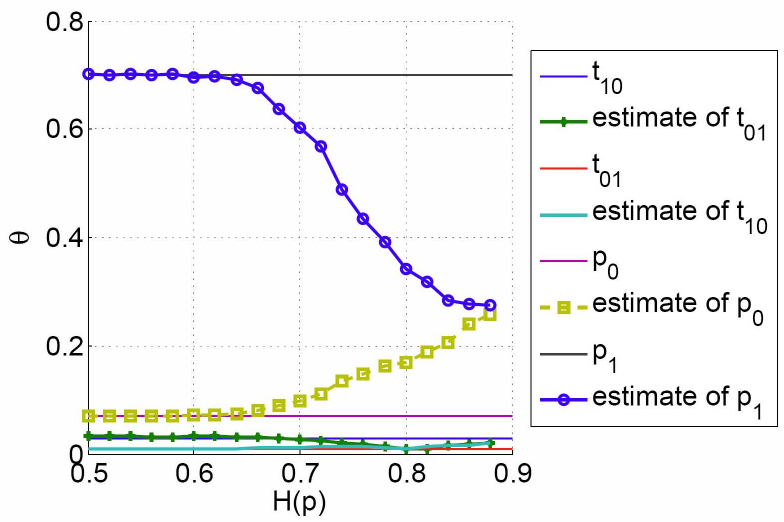

Figure 4.9: Performance of the parameter estimation, for a GE source X of parameter $\theta_X = (p_{\mathbf{0}} = 0.07, p_{\mathbf{1}} = 0.7, t_{\mathbf{10}} = 0.03, t_{\mathbf{01}} = 0.01)$.

It is shown in Fig. 4.9 that the EM algorithm manages to closely retrieve the parameters of X if the correlation is high ($H(p) < 0.65$). The more the side-information $\mathbf{y}$ is noisy (with respect to $\mathbf{x}$), the less the algorithm is able to estimate the parameters of X: our guess is that the corresponding estimated states of $\mathbf{y}$ differ too much from the actual states of $\mathbf{x}$.

Now we consider the case where the correlation channel is *predictive*, *i.e.* $X = Y \oplus Z$, and the decoders are the same ⓐ and ⓑ as before. In such a configuration, the *predictive* BSC modeling the correlation involves a *mismatch* between the true model and the one (*additive*) assumed at the decoder ⓑ. The BER of X corresponding to the *two* decoders are plotted in Fig. 4.10.

We see in Fig. 4.10 that the *mismatch* degrades the performance of the decoder ⓑ while exploiting the memory in X. In this configuration, the best decoder is the standard one, *i.e.* ⓐ, which assumes that the source is uniform, that is equivalent to assuming that the channel is *predictive*.

4.3 Syndrome-based non-asymmetric SW coding

In this Section, we design SW codes for the whole achievable rate region. Both sources are compressed in order to reach any rate on the segment between A and B (Fig. 1.1), for a given cross-over probability p. Syndrome-based approaches are known to be optimal, however maybe less amenable to rate adaptation. Yet, rate-adaptive non-asymmetric codes may be beneficial for applications such as multi-view light field coding [GD05] or for joint optimization of the rate and power of transmission in a sensor network application [RG07]. For example, in a wireless

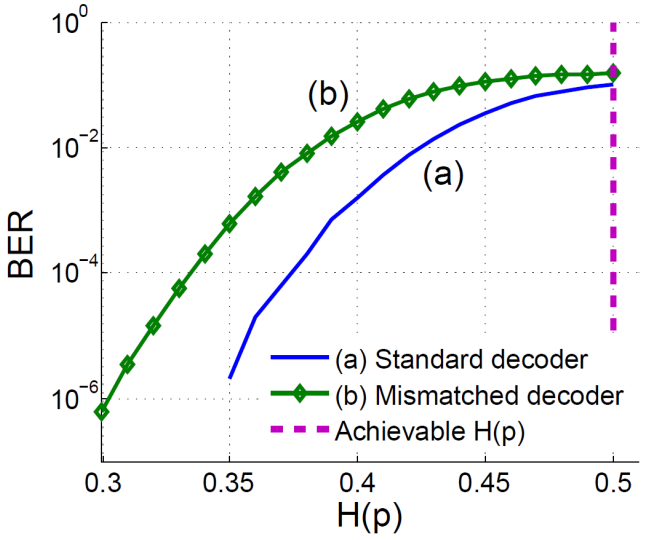

Figure 4.10: Performances of the standard decoder ⓐ and the proposed decoder ⓑ exploiting the source memory.

sensor network, spatially distributed sensors gather data and send them to a common center. It is shown in [RG07] that the optimal rate allocation depends on the transmission conditions and can therefore be any point in the SW region. It is therefore of interest to construct DSC codes that can achieve any point in the SW region. Moreover to meet the low cost constraint, each single sensor must be designed so as to handle any correlation between the sources.

In Section 4.3.1, we first consider the rate-adaptive and non-asymmetric SW coding of *uniform* Bernoulli sources, and exhibit a unique codec based on LDPC codes to achieve the whole SW bound. The approach is based on the asymmetric rate-adaptive framework developed by Varodayan *et. al* in [VAG06], and on the solution for non-asymmetric coding developed by Gehrig *et. al* in [GD05]. In Section 4.3.2, we investigate the *error propagation* phenomenon (that is induced by non-asymmetric system) by determining its causes, and we propose two solutions by modifying the matrix of the code. We finally end this study of non-asymmetric in Section 4.3.3, by adapting the codec so as to take into account the non-uniformity of the source.

4.3.1 Non-asymmetric and Rate-adaptive coding of uniform sources

To address the rate-adaptive SW problem, the authors in [VAG06] suggested to accumulate the syndrome bits of a DSC code before the puncturing step, thus maintaining the performance of the code. The proposed approach however is appropriate for the *asymmetric* setup only. In the sequel, the approach is extended in order to cover the entire Slepian-Wolf rate region, that is not limited to the *asymmetric* setup.

4.3.1.1 Non-asymmetric SW coding for a given correlation

Let X and Y be two Bernoulli sources which correlation is defined by the cross-over probability p. Let $\mathbf{x}$ and $\mathbf{y}$ be their respective realizations of length N. Let us consider an (N, K) LDPC code defined by its parity-check matrix $\mathbf{H}_{(N-K)\times N} = [\mathbf{A}_{(N-K)\times K}\ \mathbf{B}_{(N-K)\times(N-K)}]$. The syndromes $\mathbf{s_x} = \mathbf{Hx}$ and $\mathbf{s_y} = \mathbf{Hy}$, of length $(N-K)$, are computed for both sequences and transmitted to the decoder. In addition, the k' first bits of $\mathbf{x}$ ($x_1^{k'}$) and the $K-k'$ next bits of $\mathbf{y}$ ($y_{k'+1}^K$), are also transmitted to the decoder as *systematic* bits, where k' is an integer s.t. $k' \in [0, K]$. The total rates for the two sources X and Y are respectively $R_X = \frac{N-K+k'}{N}$ and $R_Y = \frac{N-k'}{N}$ bits. The structure of the coders is depicted in Fig. 4.11. Note that the particular cases $k' = 0$ and $k' = K$ correspond to the two asymmetric setups with rates corresponding to the corner points A and B of the SW region.

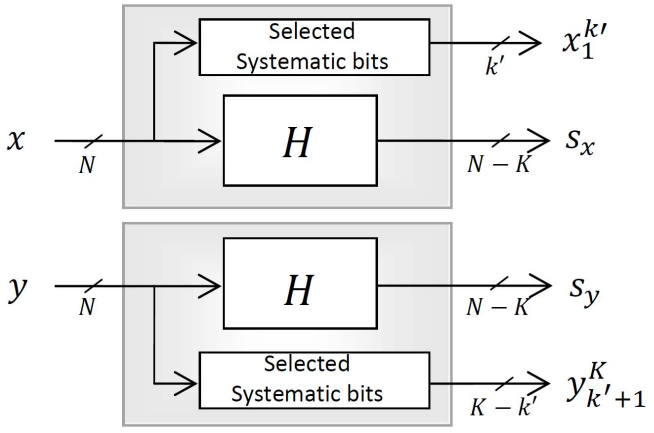

Figure 4.11: The non-asymmetric coders.

The decoder first computes $\mathbf{s_z} = \mathbf{s_x} \oplus \mathbf{s_y}$, which turns out to be the syndrome of the error pattern $\mathbf{z}$ between $\mathbf{x}$ and $\mathbf{y}$, since $\mathbf{Hx} \oplus \mathbf{Hy} = \mathbf{H}(\mathbf{x} \oplus \mathbf{y}) = \mathbf{Hz}$. A modified LDPC decoder estimates $\hat{\mathbf{z}} = \hat{\mathbf{x}} \oplus \hat{\mathbf{y}}$ from the syndrome $\mathbf{s_z}$ and the all-zero word of size N as side-information. More precisely, $\mathbf{z}$ corresponds to the vector of smallest Hamming weight with syndrome $\mathbf{s_z}$, or in other words, $\mathbf{z}$ is the vector closest to the all-zero word among the vectors with syndrome $\mathbf{s_z}$.

Once the difference pattern is found, $x_{k'+1}^K$ and $y_1^{k'}$ can be estimated thanks to the relation $\mathbf{z} = \mathbf{x} \oplus \mathbf{y}$, *i.e.* $\hat{x}_{k'+1}^K = y_{k'+1}^K \oplus \hat{z}_{k'+1}^K$ and $\hat{y}_1^{k'} = x_1^{k'} \oplus \hat{z}_1^{k'}$. The estimates of the subsequences x_{K+1}^N and y_{K+1}^N then remain to be computed. Since, $\mathbf{s_x} = [\mathbf{A}\ \mathbf{B}]\mathbf{x} = \mathbf{A}x_1^k \oplus \mathbf{B}x_{K+1}^N$, then $\hat{x}_{K+1}^N = \mathbf{B}^{-1}\ [\mathbf{s_x} \oplus \mathbf{A}\hat{x}_1^K]$, where $\mathbf{B}^{-1}$ denotes the inverse of the matrix $\mathbf{B}$ [GD05, TL05b]. Similarly, $\hat{y}_{K+1}^N$ can be calculated from $\mathbf{s_y}$, $y_{k'+1}^K$, $\hat{\mathbf{z}}$ and $\hat{x}_1^{k'}$. The following equations are a summary of these calculations:

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{x}_1^{k'} = x_1^{k'} \\ \hat{x}_{k'+1}^{K} = y_{k'+1}^{K} \oplus \hat{z}_{k'+1}^{K} \\ \hat{x}_{K+1}^{N} = \mathbf{B}^{-1}\left(\mathbf{A}\hat{x}_1^K \oplus \mathbf{s_x}\right) \end{pmatrix}$$
$$\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1^{k'} = x_1^{k'} \oplus \hat{z}_1^{k'} \\ \hat{y}_{k'+1}^{K} = y_{k'+1}^{K} \\ \hat{y}_{K+1}^{N} = \mathbf{B}^{-1}\left(\mathbf{A}\hat{y}_1^K \oplus \mathbf{s_y}\right) \end{pmatrix} \tag{4.16}$$

At this stage, we can see why a full-rank matrix $\mathbf{B}$ is mandatory for this algorithm to work. Note that for a rate K/N channel code, the parity check matrix $\mathbf{H}$ is of rank $(N-K)$, which insures that there exists a subset of $(N-K)$ columns of $\mathbf{H}$ that are linearly independent[i]; then a permutation matrix is found so that the matrix $\mathbf{H}$ has the desired form $\mathbf{H} = [\mathbf{A}\ \mathbf{B}]$.

4.3.1.2 Non-asymmetric SW coding for varying correlation

We now consider the problem of adapting the rate of the above coding/decoding system to varying correlation, by puncturing some syndrome bits. To avoid degrading the performance of the LDPC code, the syndrome bits are first accumulated before being punctured, as suggested in [VAG06]. This accumulation allows to protect the punctured syndrome bits to avoid degrading the performance of the original code. The effect of this accumulator code is equivalent to merging some rows of the parity check matrix by adding them modulo-2.

Let us consider a set of M matrices $(\mathbf{H}_m)_{m=1\ldots M}$ of respective sizes $(N-K_i) \times N, i = 1 \ldots M$ corresponding to M LDPC codes is considered. These matrices are built according to [VAG06]. Without loss of generality, assume that $\forall m, q \in [1, M], m < q \Rightarrow (N-K_m) < (N-K_q)$, meaning that the M matrices have a growing number of rows. Also consider the set of permutation matrices $(\mathbf{P}_m)_{m=1\ldots M}$ of size $N \times N$ so that $\forall m \in [1, M], \mathbf{H}_m\mathbf{P}_m$ has the requested invertible $\mathbf{B}$ part at the right place. These permutation matrices are not all the same because merging the rows of the original matrix $\mathbf{H}$ moves the location of the free columns. The coding and decoding structures are depicted in Fig. 4.12.

The coders send a first set of accumulated syndromes bits and systematic bits. With that information, the decoder tries to find a first estimate of the error pattern $\mathbf{z}$. If the decoded word $\hat{\mathbf{z}}$ does not fulfill the parity-check equations defined by $\mathbf{H}\hat{\mathbf{z}} = \mathbf{s_z}$, more syndrome bits are requested from the encoder via a one-bit feedback channel, and this goes on until $\hat{\mathbf{z}}$ has syndrome $\mathbf{s_z}$.

Let $\mathbf{H}_i$ be the matrix of rate $N : (N-K_i)$ corresponding stage i at which $\mathbf{z}$ is correctly decoded. Both the coder and the decoder know which $(N-K_i)$ columns of $\mathbf{H}_i$ are free. Let k' be an integer so that $k' \in [0, K_i]$. The systematic bits for both sources can then be sent. The source X transmits the k' bits corresponding to the k' first columns of the K_i columns of $\mathbf{H}_i$ which are *not* free. Y transmits the $K_i - k'$ bits corresponding to the $K_i - k'$ remaining *non-free* columns of $\mathbf{H}_i$. Then $R_X = \frac{N-K_i+k'}{N}$ and $R_Y = \frac{N-k'}{N}$. In other words, the parameter i first determines the

[i]The search for a subset of $(N-K)$ free columns of $\mathbf{H}$ is formally described in Annex A

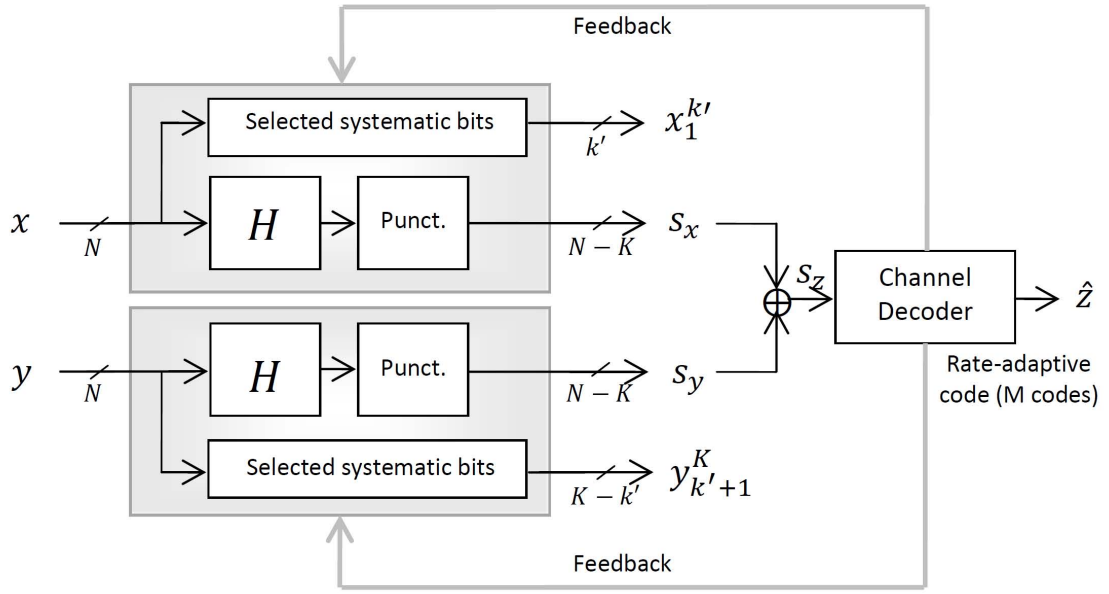

Figure 4.12: The single rate-adaptive codec.

global rate of the system, and k' fixes the specific rates of X and Y. These rates can be optimized each time a new sequence of length N is to be coded.

4.3.1.3 Simulations and results

Achieving the SW bound for any correlation parameter p

The simulation tests have been conducted with a set of matrices generated in the same way as in [VAG06]. We consider input sequences of length $N = 8052$ and generate 65 matrices having sizes ranging from 62×8052 to 8052×8052. The sources X and Y are i.i.d. random variables with uniform binary distribution. The correlation between X and Y is modeled as a BSC with crossover probability p. In our tests, p varies from 0.01 to 0.21, with a step of 0.05. For each correlation factor p, different compression rates are tested in order to achieve any point of the SW bound. This degree of freedom is obtained by tuning k', the number of input bits sent by source X. More precisely, 11 different values of k' are considered, varying from 0 to K, with a step of $0.1 \cdot K$; for each k', $2 \cdot 10^4$ words are tested.

Our system has *zero* bit error rate[ii] and we are only 0.0677 bits away from the SW bound, in average, which is among the best results reported so far for a code of that size. The results and the corresponding theoretical bounds in the Slepian-Wolf region are reported in Fig. 4.13.

Behaviour of the rate-adaptive codec

[ii]The rate-adaptive decoding is run until the decoded words are error-free

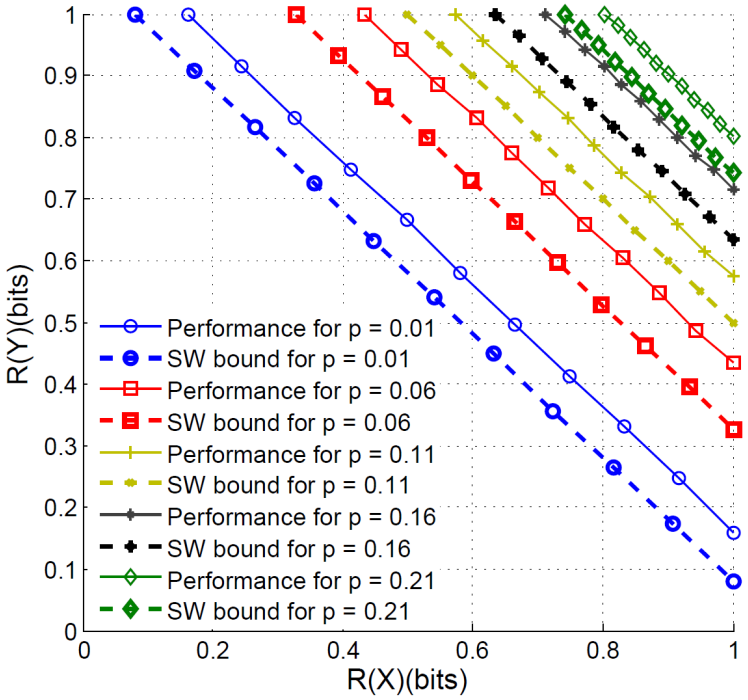

Figure 4.13: Performance of the single rate-adaptive codec.

Now, we investigate the percentage of use of each one of the M codes $(\mathbf{H}_m)_{m=1}^{M}$ in function of the correlation factor p. This is to optimize the design of each intermediate code. For example, for $p = 0.11$, we see in Fig. 4.14 that the codes of rates lower than 0.5 are rarely used, so as the codes having rates greater than 0.6; the code that is mostly used is the code of rate 0.54. Therefore, the design of rate-adaptive codes for the particular $p = 0.11$ should optimize the performance of the code of rate 0.54.

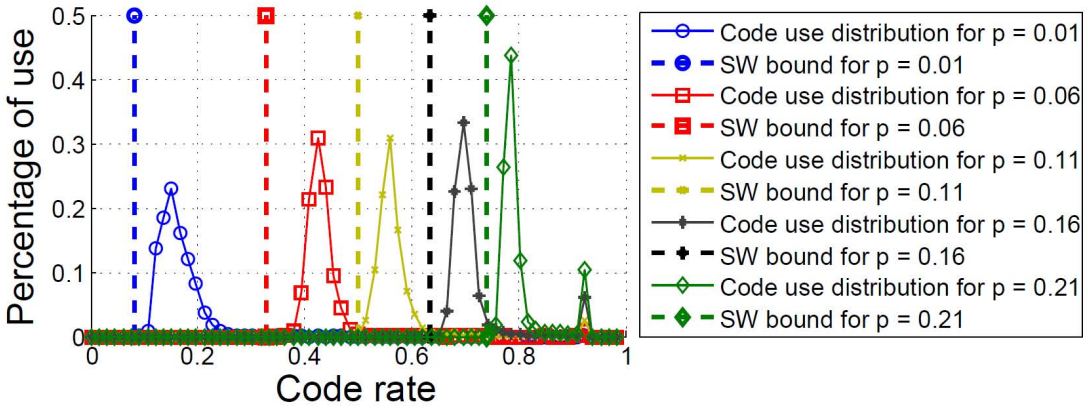

Figure 4.14: Performance of the single rate adaptive codec.

The non-asymmetric scheme presented in this Section has two drawbacks. First, if Turbo or LDPC codes are used, the inverse sub-matrix of the code, $\mathbf{B}^{-1}$, has no

periodic structure s.t. the whole matrix $\mathbf{B}^{-1}$ has to be stored, and the complexity of the multiplication grows quadratically with the block length. Second, from the construction of the decoder, we notice that the three parts of the estimated sequences $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ have different characteristics. Since the subsequences $x_1^{k'}$ and $y_{k'+1}^{K}$ are available at the decoder, their estimates have no error. The next subsequences ($x_{k'+1}^{K}$ and $y_1^{k'}$) are recovered thanks to the estimated error pattern (4.16), therefore their estimates have the same BER as the estimated error pattern. The most critical issue that we address in the next Section 4.3.2 is the BER of the $(N-K)$ unknown bits of the sequence $\hat{\mathbf{x}}$ (and similarly for $\hat{\mathbf{y}}$). Actually, multiplication by the matrix $\mathbf{B}^{-1}$ may enhance the number of errors. We call this effect the *error propagation phenomenon*. In the following, we show this effect for Convolutional and Turbo codes and we propose two solutions to limit it. Moreover, we show that the designed robust scheme is also fast in the sense that the complexity grows only linearly with the block length.

4.3.2 Error-resilient non-asymmetric SW coding of uniform sources

In this Section, we investigate the conditions under which the non-asymmetric distributed coding of uniform Bernoulli sources may prevent the *error propagation phenomenon*. We give results on the particular case of Convolution codes in Section 4.3.2.1, which gives an insight on how to implement the decoder for Turbo codes. We propose two solutions for Turbo codes. The first one, described in Section 4.3.2.2, consists in sending more systematic bits from both sources to help the decoder. This has the main drawbacks of increasing the coding rate of the code. The second solution relies on modifying the very structure of the code, to match the conditions formulated in Section 4.3.2.3, which practically consists in implementing a Turbo code having an identity matrix as the subpart $\mathbf{B}$.

4.3.2.1 Non-asymmetric SW coding using Convolutional codes

In the following, we consider a (N, K) Convolutional code. First, we note that the unknown positions in the vector $\mathbf{x}$ are design parameters. More precisely, these positions (or equivalently the columns of the matrix $\mathbf{H}$ to be extracted in order to build the matrix $\mathbf{B}$) are chosen at the encoder and known at the decoder. Therefore, the estimation of the subsequences x_{K+1}^{N} and y_{K+1}^{N} is an easier task than channel decoding over a binary erasure channel (BEC), since, in our case, the erased positions are known at the encoder.

Figure 4.15 shows the BER of the error pattern $\mathbf{z}$ (continuous line) and its effect on the estimation of the source sequence $\mathbf{x}$. The Convolutional code is defined by its parity check matrix $\mathbf{H} = \begin{pmatrix} 11 & 15 & 06 \\ 15 & 12 & 17 \end{pmatrix}$ and is punctured over a period of four trellis sections in order to get a $1:2$ compression rate for the source. Its performance is shown for three possible estimators.

First, we use the original method (Section 4.3.1.1). The dotted curve represents the BER, when Equation (4.16) is performed with an arbitrary invertible matrix $\mathbf{B}$. As expected, the BER is drastically increased. Since the linear code chosen is a Convolutional code, its parity check matrix has a natural periodic structure.

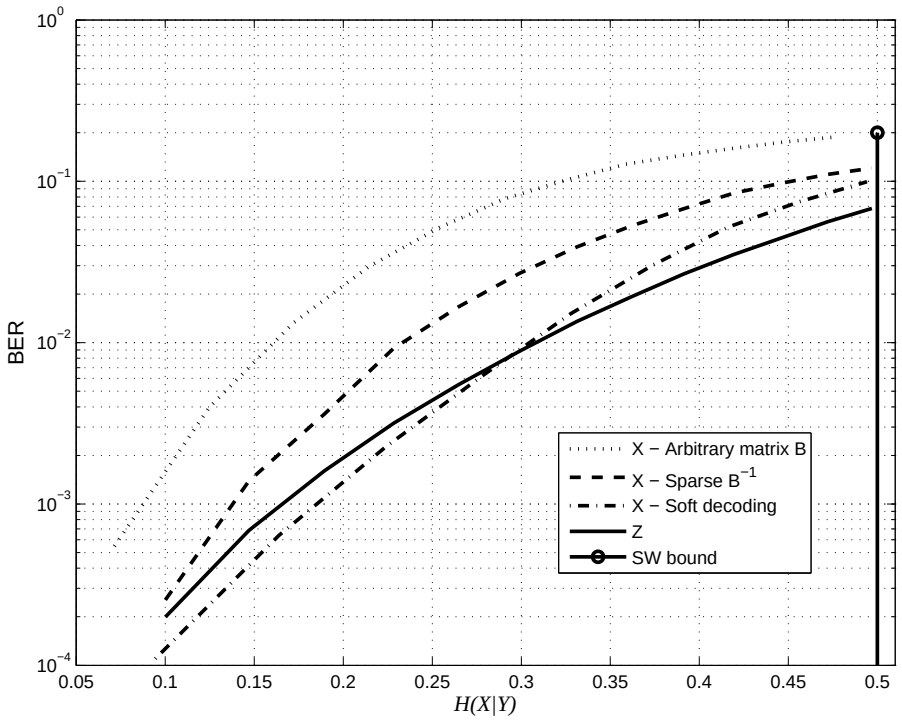

Figure 4.15: Error propagation in the estimation of the source X using a Convolutional code.

Therefore, there exists a sliding window implementation for the decoding, which reduces the required storage (a whole matrix $\mathbf{B}^{-1}$ does not need to be stored).

In a second method, we optimize the matrix $\mathbf{B}$ in order to limit the error propagation. More precisely, we choose a matrix $\mathbf{B}$ which inverse is as sparse as possible. Fig. 4.15 shows that if the matrix is chosen to be as sparse as possible, then the BER can be lowered from the dotted curve to the dashed curve. Here, an exhaustive search over all possible $\mathbf{B}$ matrices has been performed[iii].

Finally, we use a modified ML decoder to solve the problem. Let us first assume that the error pattern $\mathbf{z}$ is perfectly known at the decoder. Therefore the K first bits of $\mathbf{x}$, x_1^K, are also known. If the matrix $\mathbf{B}$ is invertible, the original source sequence is the unique sequence of the coset $\mathcal{C}_{\mathbf{s_x}}$ that has the first K bits equal to x_1^K (recall that the syndrome $\mathbf{s_x}$ is also known at the decoder). Therefore, one can build a ML decoder (Viterbi for the Convolutional decoder) that performs a search in the coset $\mathcal{C}_{\mathbf{s_x}}$ and that is matched to a channel combining a perfect channel (for x_1^K) and an erasure channel (for x_{K+1}^N). The practical effect of knowing some bits perfectly is that many wrong paths are deleted in the trellis, and, if $\mathbf{B}$ is invertible, there might be a single path that remains.

We now go back to our original problem of error propagation. If the difference pattern $\mathbf{z}$ contains errors, the decoder is matched to a channel combining a perfect channel (for $x_1^{k'}$), a BSC (for $\hat{x}_{k'+1}^K$ with cross over probability "the BER of $\mathbf{z}$") and an erasure channel (for x_{K+1}^N). Now, Fig. 4.15 shows that the BER for the estimation

[iii]The search for a subset of $(N-K)$ free columns of $\mathbf{H}$ is formally described in Annex A. The Annex also shows how to choose the optimized $\mathbf{B}$ so as to work with an inverse $\mathbf{B}^{-1}$ as sparse as possible

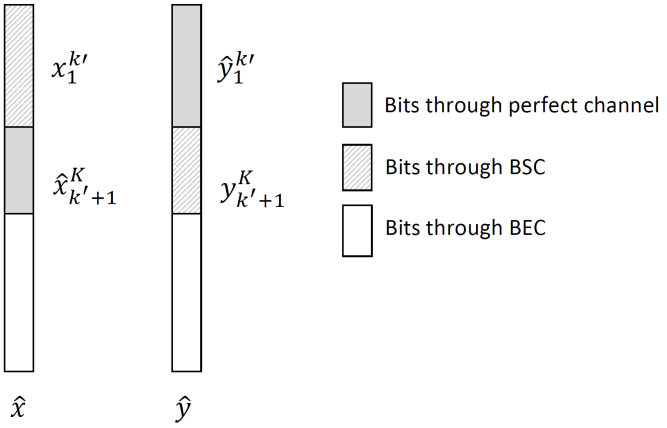

Figure 4.16: Equivalent channels for the decoding of the two sources.

of $\mathbf{x}$ is further reduced (dot-dash curve). Interestingly, with this decoder the BER of $\hat{\mathbf{x}}$ remains almost the same as that of $\hat{\mathbf{z}}$. The proposed Convolutional decoder can therefore limit the error propagation. Moreover, the complexity of the proposed algorithm grows only linearly with the block length.

4.3.2.2 Non-asymmetric SW coding with Turbo codes

In this section, we use a Turbo code composed of two identical (N, K) Convolutional codes separated by a random interleaver of size N, as shown in Fig. 4.17. Each source transmits *two* syndromes of length $(N-K)$ one of the source: $\mathbf{s_{x1}}$ (resp. $\mathbf{s_{y1}}$), and one of the interleaved version of the source: $\mathbf{s_{x2}}$ (resp. $\mathbf{s_{y2}}$).

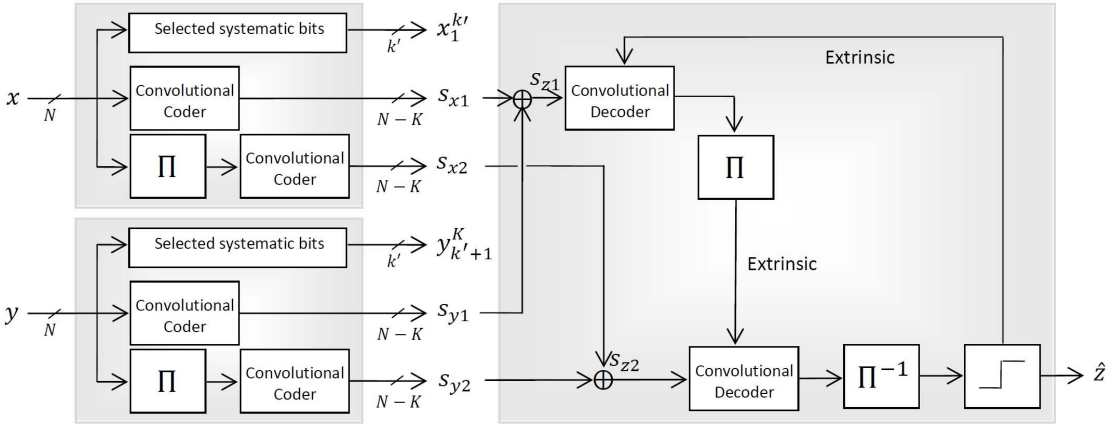

Figure 4.17: The Turbo-syndrome scheme for non-asymmetric SW coding of two sources: the encoder (left) and the estimation of the error pattern (right).

The first step of the decoding consists in estimating the error pattern $\mathbf{z}$. Therefore, we compute the two syndromes of $\mathbf{z}$: $\mathbf{s_{z}}_i = \mathbf{s_{x}}_i \oplus \mathbf{s_{y}}_i, i \in \{1, 2\}$ and search for the closest sequence to $\mathbf{0}$ in $\mathcal{C}_{\mathbf{s_{z1}}} \cap \mathcal{C}_{\mathbf{s_{z2}}}$, where $\forall i \in \{1, 2\}, \mathcal{C}_{\mathbf{s_{z}}_i}$ is the ensemble of vectors having syndrome $sv_{\mathbf{z}i}$. To perform this search, we use the modified BCJR algorithm [RLG07] for the decoding of each Convolutional code, with extrinsic message passing

between the Convolutional decoders, as depicted in Fig. 4.17. The Turbo decoding stops when the estimated $\hat{\mathbf{z}}$ matches the two syndromes $\mathbf{s}_{\mathbf{z}1}, \mathbf{s}_{\mathbf{z}2}$ or when a maximum number of Turbo decoding iterations is reached (in the sequel, we perform a maximum of 20 decoding iterations, which is a good trade-off between complexity and performance).

Once $\hat{\mathbf{z}}$ is estimated, the estimates of the source sequences $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ have to be recovered. The first method considered here consist in computing the parity check matrix of the Turbo code. Let $\mathbf{H}_1$ be the $(N-K) \times N$ parity check matrix defining the first Convolutional code. The parity check matrix of the second code $\mathbf{H}_2$ results from a permutation of the columns of $\mathbf{H}_1$. Thus, the Turbo code is completely determined by the $2(N-K) \times N$ matrix $\mathbf{H} = \binom{\mathbf{H}_1}{\mathbf{H}_2}$. Given a decomposition $\mathbf{H} = (\mathbf{A}\ \mathbf{B})$ with an invertible part $\mathbf{B}$, we apply operations described in the Equations (4.16) to estimate $\hat{x}_{K+1}^N$ and $\hat{y}_{K+1}^N$. Due to the presence of the interleaver, there exists no sliding window implementation of this estimator, contrarily to the case of Convolutional codes (unless a constrained interleaver is designed, which may degrade the performance of the Turbo code). Moreover, as expected, the error propagates drastically. Therefore, we propose a novel method that can both reduce the error propagation and have a *sliding window* implementation (to have an estimator complexity that grows linearly with N).

The estimation of the remaining subsequences proposed for Convolutional codes rely on the hypothesis that an exact MAP decoding can be used. However, for Turbo codes such a decoder is of great complexity and we consider here the usual suboptimal MAP decoder, *i.e.* the BCJR algorithm. As for the case of Convolutional codes, we first assume that the error pattern $\mathbf{z}$ has been perfectly estimated. Each constituent Convolutional decoder has to solve a linear system of $(N-K)$ equations with $2(N-K)$ unknowns and the solution is "at best" a vector subspace of dimension $(N-K)$. Let $\mathbf{B}$ be decomposed into the sub-matrices $\mathbf{B} = \begin{pmatrix} \mathbf{B}_{11}\ \mathbf{B}_{12} \\ \mathbf{B}_{21}\ \mathbf{B}_{22} \end{pmatrix}$. To insure that the dimension of each solution subspace is only $(N-K)$, the matrices $\mathbf{B}_{11}$ and $\mathbf{B}_{22}$ must be invertible. Finally, to insure that the intersection of the two solution subspaces reduces to a single sequence, the whole matrix $\mathbf{B}$ has also to be invertible.

These three new constraints (existence of $\mathbf{B}^{-1}, \mathbf{B}_{11}^{-1}$ and $\mathbf{B}_{22}^{-1}$) are enough if the two decoders communicate their solution subspaces to each other. However, to implement a low complexity decoder, we rely on the *bitwise* MAP decoding procedure that may loose some information on the solution subspace. More precisely, no information is lost if all the erased bits of a solution subspace are located in the same position, or, in other words, if a solution subspace is generated by coordinate vectors (the i-th coordinate vector has a single 1 at the i-th position and 0 everywhere else $\mathbf{e}_i = (0...010...0)$). If such a condition is not satisfied, then the Turbo decoder has to be helped by introducing some known bits at the decoder.

Figure 4.18 shows the BER of the error pattern $\mathbf{z}$ (continuous line) and its effect on the estimation of the source sequence $\mathbf{x}$. The Turbo code that we consider has a global compression rate of $1:2$. However, some unknown bits of the source are directly sent to the decoder (for both estimators of the source X). Therefore, the global coding rate (and thus the SW bound in the figure) increases to 0.54. In a nutshell, first the matrix inversion technique is used and the dashed curve shows that

the BER has been increased; then the source X is estimated with the soft decoder described above and the dot-dash curve shows that the BER has been reduced. Interestingly, our robust scheme is also fast since the complexity grows linearly with the interleaver size.

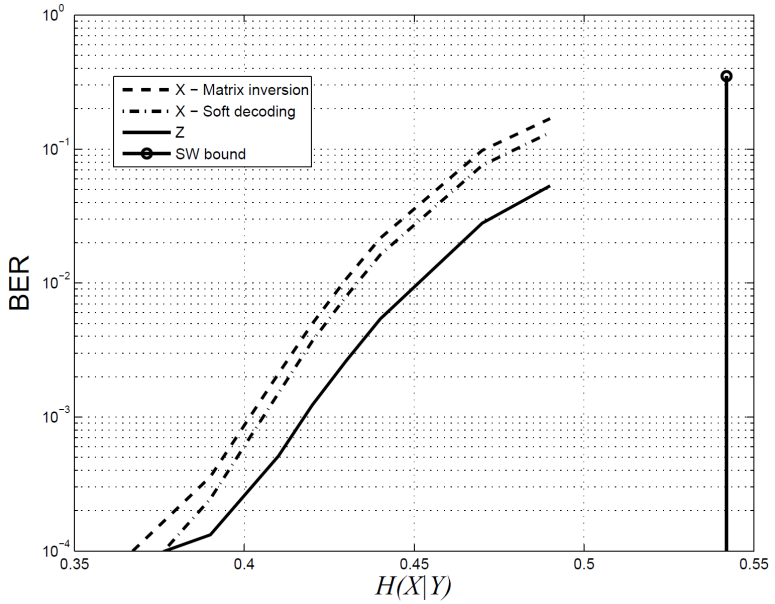

Figure 4.18: Error propagation and estimation of the source X. Each constituent code of the Turbo code is defined by $\mathbf{H} = \begin{pmatrix} 11 & 15 & 06 \\ 15 & 12 & 17 \end{pmatrix}$ and is punctured over a period of 4 trellis sections in order to get a $1:2$ compression rate. The block size is $N = 2000$.

In the following Section 4.3.2.3, we lead deeper studies to exhibit under which conditions a *maximum a posteriori* (MAP) *bitwise* decoder can recover an erased bit.

4.3.2.3 Conditions on the parity-check matrix for erasure recovery under MAP decoding

..

We take the same notation as in the previous Section. The aim is to find necessary and sufficient conditions to allow recovery of X and Y for the non-asymmetric SW coding using Turbo codes, but without adding some redundant information about the trellis states (this prevents the increase of the transmission rate), as in Section 4.3.2.2. To that end, we note that over the BEC, each bit-wise MAP Convolutional decoding has only two issues with respect to the recovery of each bit; namely, the bit is perfectly recovered (its APP is 1 or 0) or no additional information is brought after the decoding ends (its APP indefinitely remains to 0.5). Therefore, under the sub-optimal bit-wise MAP decoding of a syndrome-based Turbo code, the erased bits x_{2K-N}^{N} and y_{2K-N}^{N} can be recovered if, and only if, each matrix representation of the two Convolutional codes have a part of size $(N-K) \times (N-K)$ that is equal to $\mathbf{0}$, and the corresponding part in the matrix, of the other Convolutional code, form a basis of the subspace generated by its columns. This columns disposition is

the only possibility to insure that the bit-wise MAP decoding will output the right values of $\mathbf{x}$ and $\mathbf{y}$.

To assess the performance of the Turbo code designed according to this condition, we propose to lead some simulations using the constituent Convolutional codes defined by the party-check polynomials $\begin{pmatrix} 15/17 & 1 & 0 \\ 15/17 & 0 & 1 \end{pmatrix}$. Note that the resulting matrix of size $2(N-K) \times N$ has a part B of size $2(N-K) \times 2(N-K)$ that is the identity matrix. The performance, in terms of the BER of the system is shown in Fig. 4.19; it is compared to the performance of the codec using the matrix inversion based method for the recovery of x_{2K-N}^N and y_{2K-N}^N.

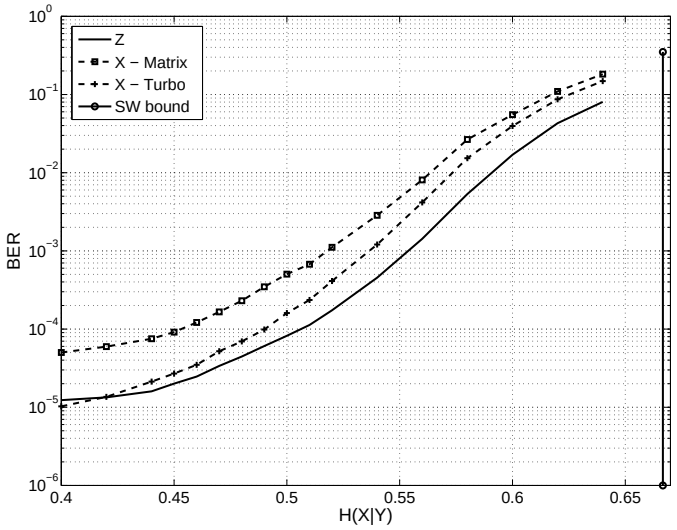

Figure 4.19: Error propagation in the estimation of the source X.

The results in Fig. 4.19 demonstrate that the designed codec limits the error propagation in the source X, and when the correlation is high enough ($H(p) < 0.42$), the remaining errors in $\hat{X}$ are even fewer than the remaining errors in $\hat{Z}$.

4.3.3 Non-asymmetric SW coding of non-uniform sources

The Bernoulli sources that are usually simulated in the literature are uniformly drawn. However, experimental sources are better described by non-uniformly drawn Bernoulli sources (see Section 5.1.1 for the modeling of the bit planes generated by a DVC system), which have a lower entropy than uniform sources. We raise the problem of *asymmetry* of the respective minimum achievable rates in the non-asymmetric SW coding of the correlated sources, due to their non-uniformity. Actually if the considered correlation channel is described by $Y = X \oplus Z$, where Z is independent of X, then the noise is *additive* for X and *predictive* for Y (see Section 3.3 for the difference between the *additive* and the *predictive* channels). This implies that $H(Y) \geq H(X)$, and *a fortiori* $H(X|Y) \leq H(Y|X)$. In Section 4.3.2, we presented the non-asymmetric coding of uniform sources using Convolutional and Turbo codes, and we derived necessary and sufficient conditions to recover the two sources under sub-optimal bitwise Turbo decoding. Here, we consider non-uniform Bernoulli sources, and we derive a necessary and sufficient condition to recover the

two sources under sub-optimal Message-Passing (MP) decoding of an LDPC code. The DSC decoder that we propose in this Section accounts for the non-uniformity of the sources, while being able to reach any point in the SW rate region.

4.3.3.1 Channel asymmetry in the non-asymmetric SW setup

In the non-asymmetric SW problem, neither X nor Y is available at the decoder: the two sources are compressed at rates R_X and R_Y fulfilling the constraints $R_X \in [H(X|Y), H(X)]$, $R_Y \in [H(Y|X), H(Y)]$, and $R_X + R_Y \geq H(X,Y)$; they have to be decoded jointly. In the literature, two non-asymmetric schemes have been proposed: in [PR00, SLXG04], the asymmetric code is partitioned into two sub-codes, one for each source, whereas the original code is used in [GD05, TL05b]. In both approaches, first the difference pattern, $\mathbf{z} = \mathbf{x} \oplus \mathbf{y}$, is estimated, then the sources are recovered. In this *two step* procedure, error propagation can occur if the error pattern $\mathbf{z}$ is not correctly estimated [HTZR09].

Since X is independent of Z, and $Y = X \oplus Z$, the BSC is additive for X and predictive for Y. Consequently, the problem is not symmetric in the two sources. When coding non-uniform Bernoulli sources, this asymmetry implies that the non-uniformity of X decreases $H(X|Y)$, but the non-uniformity of Y does not decrease $H(Y|X)$, with respect to the case where X and Y are uniform.

For the "matrix-inversion"-based non-asymmetric decoding (see Section 4.3.2), if the decoding of $\mathbf{z}$ is successful, the matrix-inversion-based decoding is able to retrieve the original values of the sources since $\mathbf{B}$ is invertible. Anyhow, this matrix-inversion method has some limitations. More precisely:

- It does not exploit the non-uniformity of the sources, so the SW bound of X is not reached;
- It suffers from the *error propagation* phenomenon reported in Section 4.3.2, since the decoding cannot take into account the uncertainty from the decoding of $\mathbf{z}$;
- It does not allow to take into account the BSC type (additive or predictive) between the two sources .

The proposed LDPC-based non-asymmetric decoder is described in the following Section. We have designed it so as to take into account the non-uniformity of the sources. The error propagation phenomena is also dealt with, since the decoding is not dependent on the decoding of the error pattern $\mathbf{z}$. Finally, the type of the BSC between the sources is also taken into account.

4.3.3.2 The proposed non-asymmetric decoding

The decoder must find the best estimates $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ given their syndromes $\mathbf{s_x}$, $\mathbf{s_y}$, their systematic parts $x_1^{k'}$, $y_{k'+1}^K$, and knowing *a priori* that X and Y are non-uniform binary sources with respective parameters p_X and p_Y. This amounts to solving the joint bitwise maximization problem: $\forall n \in [1, N]$

$$\begin{cases} \hat{x}_n = \arg \max\limits_{x \in \{0,1\}} \mathbb{P}(X_n = x | x_1^{k'}, y_{k'+1}^K, \mathbf{s_x}, \mathbf{s_y}) \\ \hat{y}_n = \arg \max\limits_{y \in \{0,1\}} \mathbb{P}(Y_n = y | x_1^{k'}, y_{k'+1}^K, \mathbf{s_x}, \mathbf{s_y}) \end{cases} \tag{4.17}$$

The computation of the *a posteriori* probabilities (APP) for this problem is too complex, but it can be approached with a sub-optimal MP algorithm. We now present the graph that shows the factorization of the APP from the decoding.

Graph for the decoding algorithm

In the context of DSC, an (N, K)-LDPC code $\mathcal{C}$ can be represented either by its *parity-check matrix* $\mathbf{H} = (h_{mn})$ of size $(N - K) \times N$ or by a *bipartite graph*. The bipartite graph is composed of $(N - K)$ check nodes and N variable nodes, representing respectively the syndrome symbols and the source symbols. $h_{mn} = 1$ if the m-th check node is connected to the n-th variable node. Moreover, the correlation between the sources X and Y is modeled by a check node called the *BSC node* and represents the modulo-2 sum $Y = X \oplus Z$. The graph that describes all the dependencies among the variables is shown in Fig. 4.20.

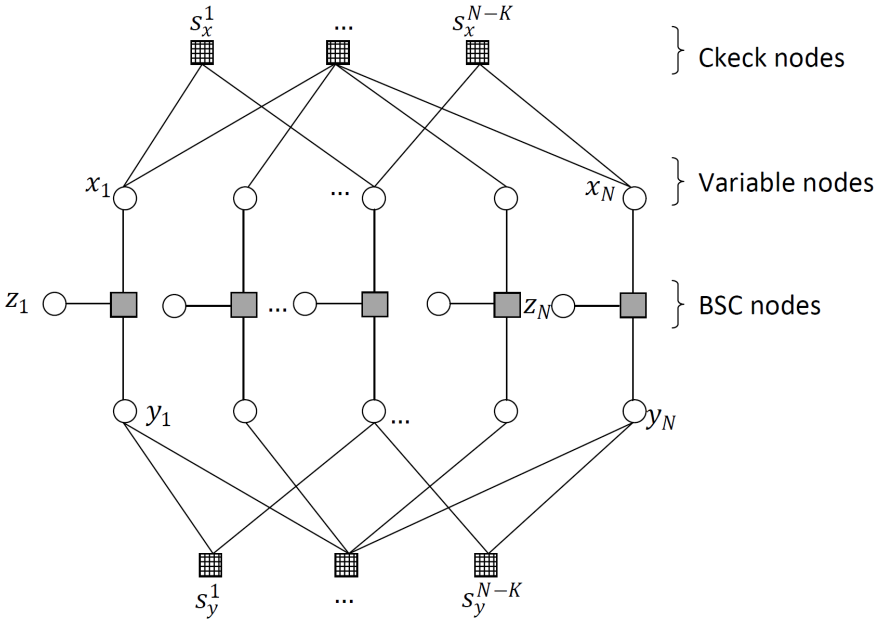

Figure 4.20: Joint graph describing the *joint* Slepian-Wolf decoding.

Definition of the messages

Consider the following notation and definition for the messages that are passed in the graph for the decoding procedure.

- $I(x_n)$ (resp. $I(y_n)$, $I(z_n)$): *intrinsic* information for the node x_n (resp. y_n, z_n);
- $E_{n,e}, e \in [1, d_{xn}]$: messages passed from the variable nodes, on their e-th edge, to the check nodes;
- $Q_{m,e}, e \in [1, d_{sm}]$: messages passed from the check nodes, on their e-th edge, to the variable nodes;

- $V(x_n)$: messages passed from the variable nodes to the BSC nodes;
- $B(x_n)$ (resp. $B(y_n)$): messages passed from the BSC nodes to the variable nodes x_n (resp. y_n).

All the messages are Log-Likelihood Ratio (LLR). They are labeled (in) or (out) if they respectively come *to* or *from* the considered node.

Update rules Now, we describe the messages that are passed between the nodes of the graph, and show how they are updated through the joint sum-product decoding, for the particular case of non-uniform sources. The factorization of the probabilities in Equation (4.17) shows that:

Intrinsic information computation

$$I(x_n) = \begin{cases} (1-2x_n)\cdot\infty, \forall n \in [1,k'] \\ \log\left(\dfrac{1-p_X}{p_X}\right), \forall n \in [(k'+1),N] \end{cases}$$
$$I(y_n) = \begin{cases} 0, \forall n \in [1,k'] \cup [(K+1),N] \\ (1-2y_n)\cdot\infty, \forall n \in [(k'+1),K] \end{cases}$$
$$I(z_n) = \log\left(\frac{1-p}{p}\right)$$

Messages from the variable nodes to the check nodes

$$E_{n,e}^{(out)} = I(x_n) + \sum_{k=1,k\neq e}^{d_{xn}} E_{n,k}^{(in)} + B(x_n) \tag{4.18}$$

Messages from the check nodes to the variable nodes

$$Q_{m,e}^{(out)} = 2\tanh^{-1}\left[(1-2s_n)\prod_{k=1,k\neq e}^{d_{sm}} \tanh\frac{Q_{m,e}^{(in)}}{2}\right] \tag{4.19}$$

Messages from the variable nodes to the BSC nodes

$$V(x_n) = I(x_n) + \sum_{k=1}^{d_{xn}} E_{n,k}^{(in)} \tag{4.20}$$

Messages from the BSC nodes to the variable nodes

$$B(x_n) = 2\tanh^{-1}\left[\tanh\left(\frac{V(y_n)}{2}\right)\tanh\left(\frac{I(z_n)}{2}\right)\right]$$
$$B(y_n) = 2\tanh^{-1}\left[\tanh\left(\frac{V(x_n)}{2}\right)\tanh\left(\frac{I(z_n)}{2}\right)\right]$$

As the expressions of E, Q and V are similar for X and Y, only the update rules for X have been described in the equations (4.18), (4.19), (4.20) above.

4.3.3.3 Condition for the recovery of both sources

The soft decoding algorithm presented in Section 4.3.3.2 above does not perform exact computation of the APP of $\hat{\mathbf{x}}$. Instead, the proposed sub-optimal MP algorithm may not be able to perfectly recover the source symbols. In the following, we derive necessary and sufficient conditions to recover the sources.

Let us first consider the *asymmetric* case. K symbols of a source (say X) as well as its $(N-K)$ syndrome bits and the $(N-K)$ syndrome bits of the source Y, are available. First, the remaining $(N-K)$ symbols of X, corresponding to the part $\mathbf{B}$ of $\mathbf{H}$, have to be decoded by a BEC MP algorithm. This is similar to the problem of solving a linear system of $(N-K)$ equations with $(N-K)$ unknowns. In that case, the following Lemma stands:

Lemma 4.1. *Let $\mathbf{x} \in \mathbb{F}_2^N$ satisfy a set of $(N-K)$ linear equations defined by a matrix $\mathbf{H}$ of size $(N-K) \times N$ s.t. $\mathbf{Hx} = \mathbf{s_x}$. Assume that the first K bits of $\mathbf{x}$ (denoted x_1^K) are known and that the last $(N-K)$ bits (x_{K+1}^N) are unknown. The system of linear equations can be rewritten as $\mathbf{Hx} = [\mathbf{A}\ \mathbf{B}][x_1^K\ x_{K+1}^N]^T$ where $\mathbf{B}$ is assumed to be an invertible square matrix. A necessary and sufficient condition to recover the unknowns, with the BEC message-passing (MP) algorithm, is that $\mathbf{B}$ is triangular (up to a permutation of its columns).*

Proof. First, recall that the BEC MP algorithm is equivalent to greedily checking whether any of the parity-constraints can solve an yet unknown value from already known ones. It follows that the condition is *sufficient*. Conversely, assume that we can recover the unknowns under the BEC MP algorithm. To start the process, there must be at least one equation with *only one* unknown. Moreover there is at most one equation with the same unknown (if not, $\mathbf{B}$ would not be invertible). This unknown can therefore be recovered and we now have to solve a system of $(N-K-1)$ equations with $(N-K-1)$ unknowns. The rest of proof follows by induction since, at each step of the algorithm, the number of equations and the number of unknowns are simultaneously reduced by one. Therefore, the condition is also *necessary*. □

Lemma 4.1 derives a necessary and sufficient condition for the exact recovery of the source X. The second source Y can now be recovered since its syndrome is known and since the source X is available. Therefore Lemma 4.1 derives the necessary and sufficient condition for the estimation of the sources X and Y in the *asymmetric* case, when decoded with the proposed algorithm presented in Section 4.3.3.2.

We now consider the *non-asymmetric* SW problem. Note that the *asymmetric* case is the best case scenario of the *non-asymmetric* one. Thus, the condition in Lemma 4.1 becomes a *necessary* condition for the recovery of the two sources.

4.3.3.4 Experimental setup and simulation results

Code design

According to Lemma 4.1, the invertible part $\mathbf{B}$ of the channel code's parity-check matrix has to be triangular in order to ensure the decoding. To design our LDPC code, we take ideas from the design of Irregular Repeat-Accumulate (IRA) codes [JKM00], and we follow the design rules presented in [JW05]. It imposes the shape in Fig. 4.21 to the parity-check matrix, with a pattern of weight-two columns forming two diagonal lines of ones, and a pattern of weight-three columns ensuring a good degree distribution for the code. The columns forming the weight-two and weight-three columns are chosen as the triangular part $\mathbf{B}$.

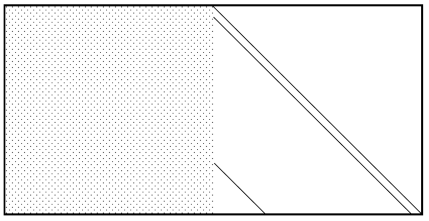

Figure 4.21: Shape of the designed LDPC code's parity-check matrix, inspired by the structure of IRA codes.

That structure is the best way to impose structured weight-two and weight-three columns without adding cycles of short lengths amongst the variable nodes. We take the concatenation of these columns as our part $\mathbf{B}$. We generate a rate-$\frac{1}{2}$ LDPC code of block length 1000. More precisely, we first find the optimal *variable degree distribution* $\Lambda(x)$, and the optimal *check degree distribution* $\Phi(x)$, of a rate-$\frac{1}{2}$ LDPC code by density evolution [RSU01]. Then, the LDPC code is obtained using the Progressive Edge Growth (PEG) principle [HEA05] by imposing the structure shown in Fig.4.21 to the generated parity-check matrix (such a code generation is described with more details in Annex B).

The code we have generated has the following variable degree distribution: $\Lambda(x) = 0.483949x + 0.294428x^2 + 0.085134x^5 + 0.074055x^6 + 0.062433x^{19}$. Note that for this rate-$\frac{1}{2}$ LDPC code, the proportion of variable nodes of degree two and three $\Lambda_2 + \Lambda_3 = 0.7784 \geq 0.5$ ensures that there are enough weight-two and weight-three columns to form the part $\mathbf{B}$ that we need. The check degree distribution is $\Phi(x) = 0.741935x^7 + 0.258065x^8$. Note that the choice of these degree distributions do not depend on the source statistics. The same code is used for all the simulations presented in the present Section.

Performance of the non-asymmetric SW codec

Now, we turn to comparing the performance of our codec and the matrix-inversion-based codec for different values of p. The source X is non-uniform, with $p_X = 0.12$, Z is a BSC of cross-over probability p, and Y is obtained by $Y = X \oplus Z$. For each value of p, 10 values of k' are considered, and for each value of k', $2 \cdot 10^4$ realizations $\mathbf{x}$ are tested, the obtained BER have been averaged over all the values of k'.

For a non-uniform source $X \sim \mathcal{B}(p_X = 0.12)$, $H(X|Y) = 0.5$ occurs for $H(p) = 0.93$, whereas $H(Y|X) = 0.5$ occurs for $H(p) = 0.5$. This difference comes from the BSC between the sources which is not symmetric in the two sources. The performance of *four* systems are presented in Fig. 4.22:

- ① Asymmetric codec, for the decoding of X, with *additive* BSC as correlation channel.
- ② Asymmetric codec, for the decoding of Y, with predictive BSC.
- ③ Non-asymmetric codec, with the matrix inversion [GD05]; ③a for the decoding of X and ③b for the decoding of Y.
- ④ Non-asymmetric codec, proposed in this Section; ④a for the decoding of X and ④b for the decoding of Y.

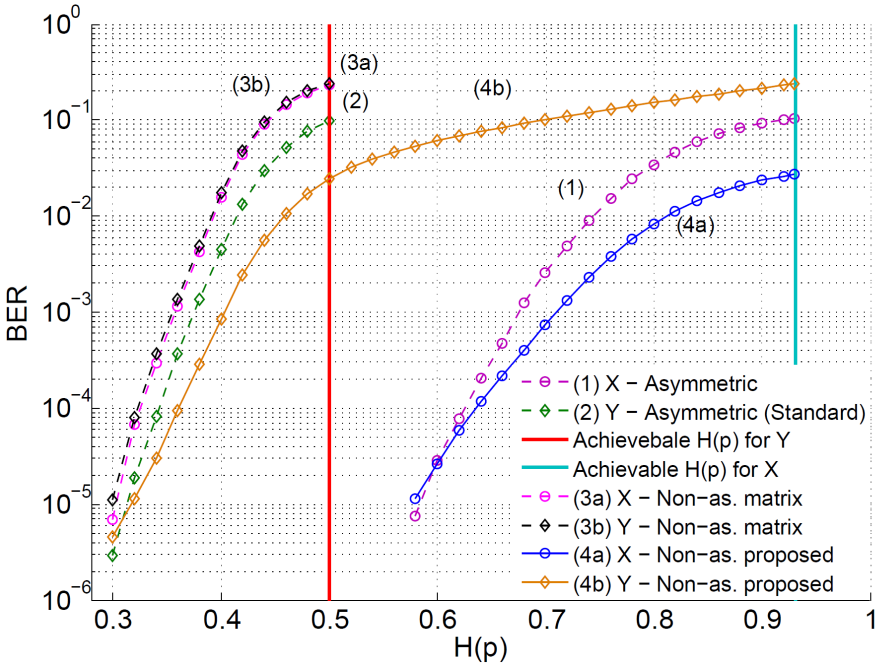

Figure 4.22: Performance of the non-asymmetric SW codec, for a non-uniform source $X \sim \mathcal{B}(p_X = 0.12)$.

The decoding involving the matrix inversion is the worst solution, since no measures are taken to prevent the error propagation between the step of estimating $\hat{Z}$ and the decoding of the sources, that is why its BER in the non-asymmetric setup is worse than the BER in the asymmetric one. The joint method that we propose, with cross message passing during the decoding of the two sources, outperforms the matrix-inversion based method. The statistics of the sources are taken into account for the decoding, as well as the BSC models between the sources.

Behavior analysis of the non-asymmetric SW codec in function of k'

In this section, we compare the behavior of our codec and the behavior of the matrix-based codec in function of k', for a given value of $H(p) = 0.62$ ($p = 0.1541$). 10 values of k', varying from 0 to K, are considered; and, for each value taken by k', $2 \cdot 10^4$ realizations of X are tested. The obtained BER are plotted in Fig. 4.23 in function of the ratio $\frac{k'}{K}$.

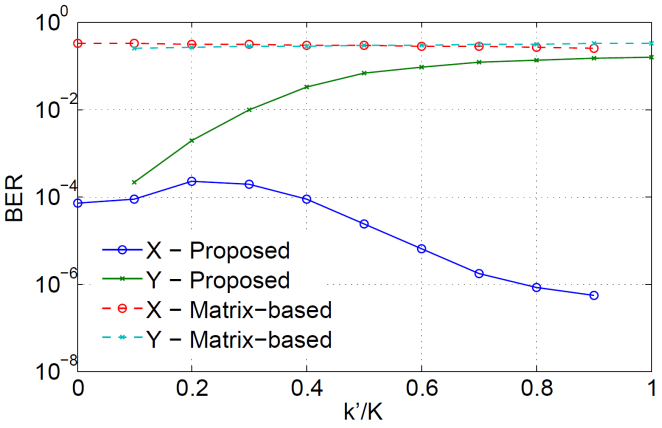

Figure 4.23: Performance of the non-asymmetric SW codec in function of $\frac{k'}{K}$, for a non-uniform source $X \sim \mathcal{B}(p_X = 0.12)$ at $H(p) = 0.62$.

The results presented in Fig. 4.23 confirm that the joint decoding of the non-uniform sources X and Y is *not symmetric* in the two sources, as claimed in Section 4.3. Indeed, if the source X is decoded with a very low BER for all the values of k', but the source Y is only well decoded when $k' = 0$ (which corresponds to the asymmetric case). Besides, the decoding of the two sources is not symmetric in k', and their BER is not equal even when $\frac{k'}{K} = 0.5$. That comes from the non-uniformities of the sources that are not equal: if $p_X = 0.12$ is constant, p_Y depends on p and is given by $p_Y = p_X(1-p) + p(1-p_X)$. Finally, we note that the matrix-inversion based decoding is symmetric in the two sources and in k', since it does not take into account the source probabilities nor the type of BSC (additive or predictive).

4.4 Conclusion

As pinpointed in Chapter 3, the achievable SW bound for the coding of binary sources which distribution is not uniform is lower than that of uniform sources. Namely, for the asymmetric SW coding problem, we have designed some novel tools, presented in this Chapter, that allow to achieve the asymmetric SW bounds for non-uniform Bernoulli sources and Gilbert-Elliott sources. We showed that the changes to add to the existing Turbo and LDPC decoders are minimal, with respect to the rate gain that is expected, compared to the rate of uniform sources. Also presented in this Chapter are the single codecs that allow to reach the whole non-asymmetric SW bound, for any correlation p. Finally, we combine the two concepts to reach any point on the SW bound, for a source that is non-uniformly distributed. The

main issue that we encountered is the asymmetry of the achievable rates for the two sources, which comes form the BSC model that can be *additive* or *predictive* depending on the source; this issue has been solved by adapting the message update rules for the joint decoding.

Chapter 5

Contributions to Distributed video coding

In this Chapter, the models and tools, that we presented in Chapters 3 and 4 and validated for theoretical and synthetic sources, are integrated in a transform-domain Distributed Video Coding (DVC) system. They are shown to enhance the codec by improving its rate-distortion performance. First, in Section 5.1, the non-uniform source model is shown to be a better model than the uniform one for the bit planes generated by the DVC codec. The integration of the LDPC-based estimation-decoding algorithm of Section 4.1.2 is then shown to improve the rate-distortion performance of the codec by up to 5.7% with respect to the standard one. Then, in Section 5.2, the bit planes are modeled as GE memory sources. This model is shown to be an even better model than the non-uniform one, and we present experimental results that prove the efficiency of the proposed method. The rate-distortion performance of the codec is improved by up to 10.14%. The two models are finally compared to the Markov source model, which is revealed to be unadapted for the video bit planes. For these three source models, one has to deal with an intrinsic problem, that is the correlation model between the WZ bit planes and the side-information (*additive* or *predictive*). In this Chapter, we give the rules to choose between these two correlation models.

5.1 Non-uniform source model for DVC

In Section 3.1, we have investigated the non-uniform Bernoulli source model and we established that a considerable gain was expected for the coding, when the channel between the source and the side-information is *additive* and when the source distribution is correctly exploited. In this Section, we model the Wyner-Ziv (WZ) bit planes that are generated by the DVC codec DISCOVER as non-uniform sources which parameters must be estimated on-line. First, we investigate the accuracy of the modeling, and we describe how to use the decoding described in Section 4.1.2 in the DISCOVER codec. Finally, we show some experimental results for the coding of video sequences that prove the pertinence of the modeling.

5.1.1 Accuracy of the non-uniform source modeling

We investigate the distributions of the bit planes coded by the DISCOVER codec, by assessing *off line* their *Bernoulli parameters*. We show in Fig. 5.1 the probability of 1's in some WZ bit planes from the five QCIF video sequences *Hall monitor, Foreman, Coastguard, Flower*, and *Soccer* [i].

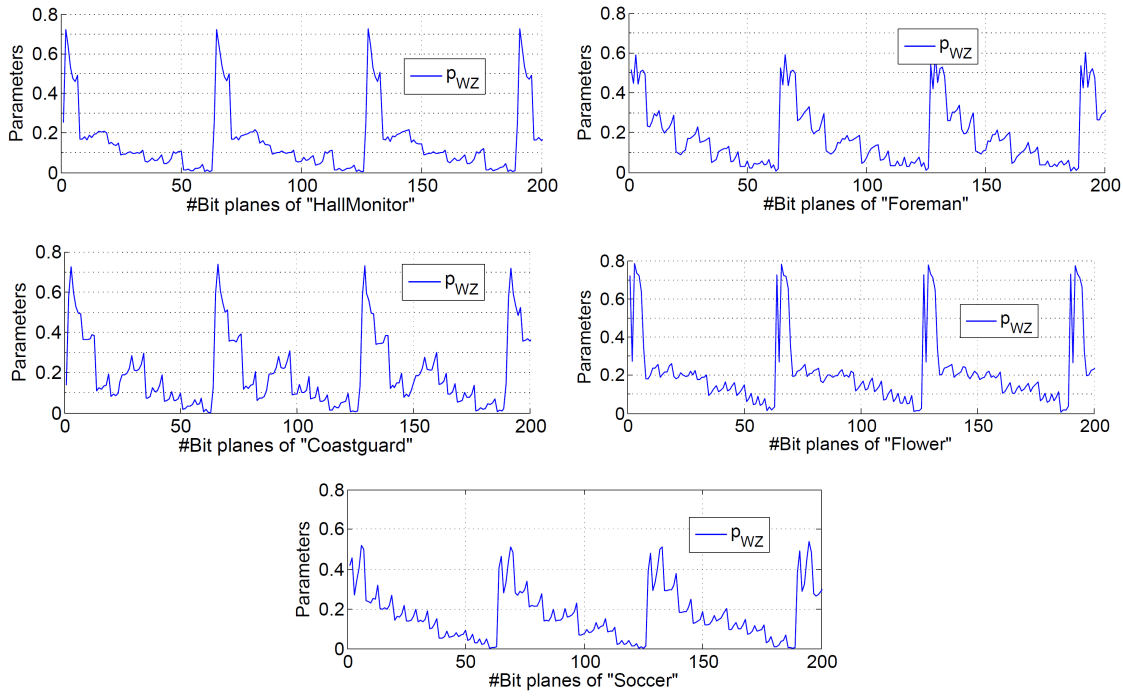

Figure 5.1: Probability of 1's in the bit planes taken individually.

The results on Fig. 5.1 confirm that the bit planes of the five video sequences are non-uniformly distributed, besides their distribution varies from bit plane to bit plane. This justifies that the non-uniform modeling is a better model than the uniform one, and that we have to estimate the parameter of each bit plane individually.

5.1.2 Implementation and experimental results

5.1.2.1 Upgrade of the LDPC decoding for non-uniform sources

To use the decoding presented in Section 4.1.2, we modify the intrinsic computation (4.2) to take into account the correlation channel in DISCOVER. The Laplacian channel models the correlation between the generated WZ DCT coefficients $\mathbf{X}_k$ and the SI DCT coefficients $\mathbf{Y}_k$, i.e. $\mathbf{Y}_k = \mathbf{X}_k + \mathbf{Z}_k$, where $\mathbf{Z}_k$ stands for the Laplacian noise, and k designates the frequency band. The density function of the noise is given by $p_{Z_k}(z) = \frac{\alpha_k}{2} e^{-\alpha_k |z|}$, where α_k is the Laplacian parameter estimated from each frequency band of the SI. That channel comprises the information from the previously decoded bit planes in the current frame. More precisely, $\forall n \in [1, N]$,

[i] One picture of these five video sequences are shown in Annex C

for each bit $x_{k,n}^b$, of each bit plane b, of each DCT frequency band k, the channel "cross-over" probability $p_{k,n}^b$ is computed as:

$$p_{k,n}^b = \mathbb{P}\left(X_{k,n}^b = 1 | y_{k,n}, x_{k,n}^1, \ldots, x_{k,n}^{b-1}\right) = \frac{\int\limits_{x \in Q(1)} \frac{\alpha_k}{2} e^{-\alpha_k |y_{k,n} - x|} dx}{\int\limits_{x \in Q(1) \cup Q(0)} \frac{\alpha_k}{2} e^{-\alpha_k |y_{k,n} - x|} dx}$$

where

- $y_{k,n}$ is the n-th DCT coefficient of the frequency band k;
- $Q(m) = Q(m, x_{k,n}^1, \ldots, x_{k,n}^{b-1})$ is the set of all the quantization intervals corresponding to the quantized symbols, which b most significant bits are $(x_{k,n}^1, \ldots, x_{k,n}^{b-1}, m)$.

To use the adapted LDPC decoder in the DISCOVER codec, Equation (4.2) of the original decoding becomes:

$$I_n(p_X) = \log\left(\frac{\mathbb{P}(X_n = 0|y_n)}{\mathbb{P}(X_n = 1|y_n)}\right) = \begin{cases} \log\left(\frac{1 - p_{k,n}^b}{p_{k,n}^b}\right), \; if \; the \; BSC \; is \; predictive \\ \log\left(\frac{1 - p_{k,n}^b}{p_{k,n}^b}\right) + \log\left(\frac{1 - \hat{p}_{WZ}}{\hat{p}_{WZ}}\right), \; if \; additive \end{cases} \tag{5.1}$$

where $\hat{p}_{WZ}$ is initialized with $\hat{p}_{SI}^b$, the probability of 1's in the binarized version of the SI. The estimate $\hat{p}_{WZ}$ is updated through the iterations according to the update rule of the EM algorithm given by Equation 4.3.

5.1.2.2 Minimum rate estimation for rate-adaptive decoding of non-uniform sources

The minimum rate for the rate-adaptive decoding needs to account for the non-uniformity of the source, given that the correlation channel can be additive or predictive. First, if the correlation is predictive, then the minimum rate is given by Equation (2.7) which is also the minimum rate for uniform sources. Now, consider that $\forall b, \forall i, \forall k$, $x_{k,i}^b$ is the realization of a non-uniform Bernoulli source of parameter p_X. Therefore, $\forall t \in \{0, 1\}$, the crossover event in Equation (2.5) becomes:

$$\mathbb{P}\left(x_{k,i}^b = t | y_{k,i}^b, x_{k,i}^1, x_{k,i}^{b-1}\right) = \mathbb{P}\left(y_{k,i}^b, x_{k,i}^1, x_{k,i}^{b-1} | x_{k,i}^b = t\right) \mathbb{P}\left(x_{k,i}^b = t\right) \tag{5.2}$$

In Equation (5.2), the term $\mathbb{P}\left(y_{k,i}^b, x_{k,i}^1, x_{k,i}^{b-1} | x_{k,i}^b = t\right)$ has the same value as the term $\mathbb{P}\left(x_{k,i}^b = t | y_{k,i}^b, x_{k,i}^1, x_{k,i}^{b-1}\right)$ in Equation (2.5) for uniform sources, and the term $\mathbb{P}\left(x_{k,i}^b = t\right)$ is equal to p_X if $t = 1$, and $(1 - p_X)$ if $t = 0$. This expression brings to an equivalent crossover probability p_{cr}^{NU} that is further from 0.5 than the original p_{cr} in Equation 2.6 (NU stands for "Non-Uniform"). Thus the minimal achievable rate $R_{min}^{NU} = H(p_{cr}^{NU})$ is lower than the original $H(p_{cr}$ corresponding to uniform sources.

Since the channel between the WZ bit plane and the side-information (SI) is modeled by a complex channel, we cannot assess on-line the additive or predictive nature of the correlation. Namely, we cannot establish a relation between the entropies of the WZ bit planes and the SI as needed in Claim (3.1). Therefore, we make two decodings using *both* channel models, and we choose *a posteriori* the one (additive or predictive) that yields the best result.

5.1.2.3 The channel is assumed to be additive

We first assume that the channel between the WZ bit planes and the SI is only additive. The improvement brought by non-uniform source modeling is presented in Fig. 5.2 and in Table 5.1 (for the highest PSNR only) for the five video sequences *Hall monitor, Foreman, Coastguard, Flower*, and *Soccer*.

The rate gain occurs only for the sequence *Soccer*; it is $1.88kbps$ (0.65%). This low rate gain contrasts with the huge non-uniformity of the bit planes (Fig. 5.1). In view of these results, we admit that the channel modeling the correlation is *not always* additive and has to be dynamically determined by the decoder.

Sequence	Rate loss (kbps)	Rate loss (%)
Hall Monitor	−9.58	−10.28
Foreman	−7.86	−3.59
Coastguard	−14.85	−8.35
Flower	−17.79	−8.81
Soccer	1.88	0.65

Table 5.1: Rate gain for the five video sequences, when the correlation is assumed to be only additive.

5.1.2.4 The channel is unknown and assessed by the decoder

As the correlation model between the WZ and SI frames is sometimes *predictive* and sometimes *additive*, we first perform the decoding of each WZ bit plane with the *predictive* assumption (here, this corresponds to the decoding that is already performed by the standard decoder). If this decoding process fails (the criteria used are presented in Section 4.2.6), then we perform a new decoding with the *additive* assumption to exploit the non-uniformity. Our *a posteriori criterion* to choose between the predictive and the additive model is the decoding failure under the predictive assumption, while the decoding under the additive assumption is successful. The decoder improvement corresponding to this adaptive approach is shown in Fig. 5.3.

Our decoder improves the rate of the WZ frames by an interesting amount; the decrease is up to $16.57kbps$ (5.7%) for the sequence *Soccer* at the highest PSNR. The rate gain is presented in Table 5.2. That decrease proves that it is worth taking into account the non-uniformity of the bit planes, as long as the channel model is additive.

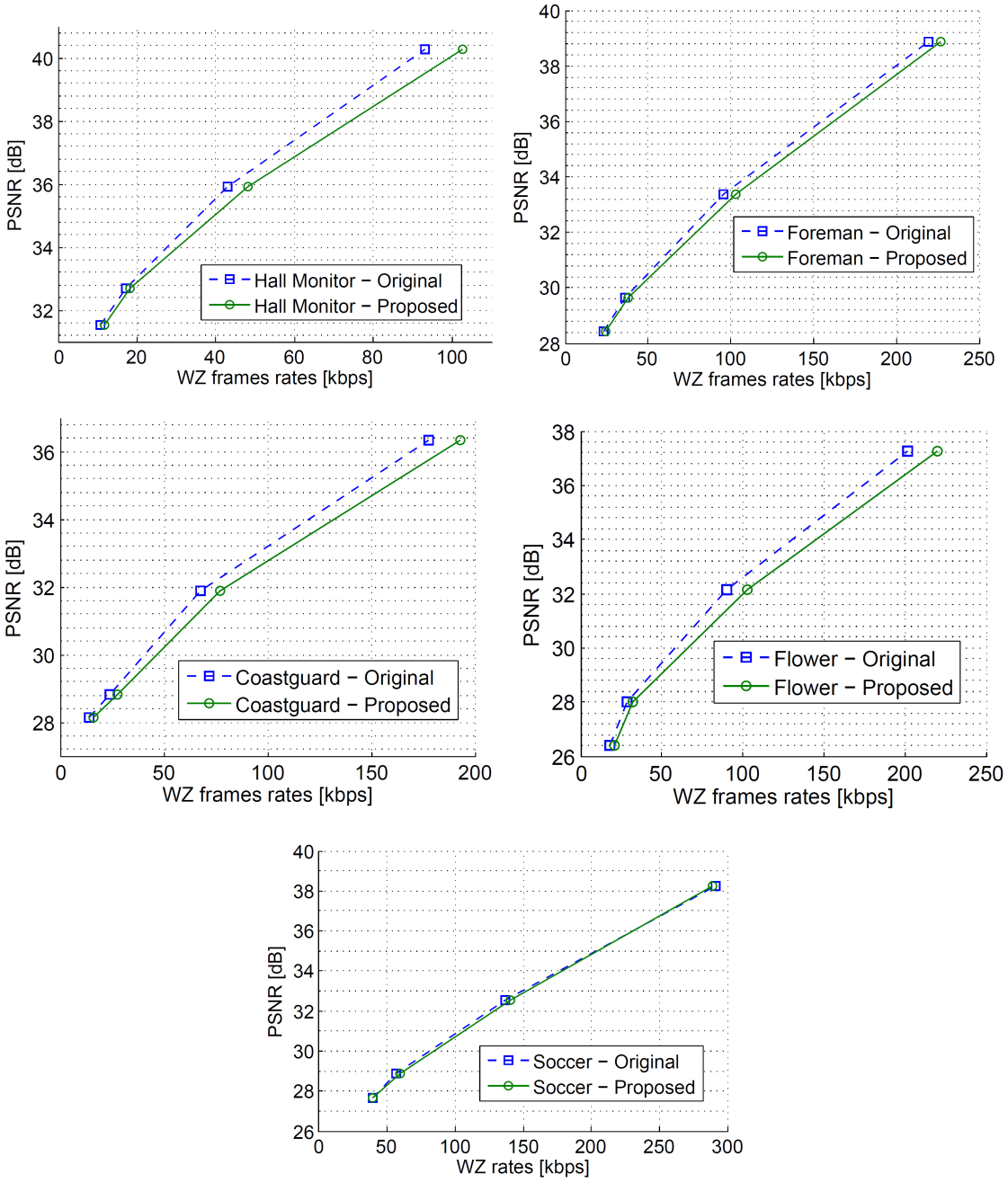

Figure 5.2: Over an additive channel, the proposed decoder is sometimes worse than the standard one to render the videos at the same PSNR.

One may wonder why we have not used the Bernoulli parameter estimator of Section 3.4 to guess the WZ bit planes distribution before the decoding. The answer lies in the rate-adaptive framework tat is adopted in the DISCOVER codec: when the syndrome bit number is too small, the parameter estimation may give false values since there is not enough syndrome realization for the estimator described in Equation (3.18) to be trusted in the first place.

As a conclusion to this Section, we stress the fact that the non-uniformity of the bit planes is not a sufficient condition to decrease their compression rates. Actually,

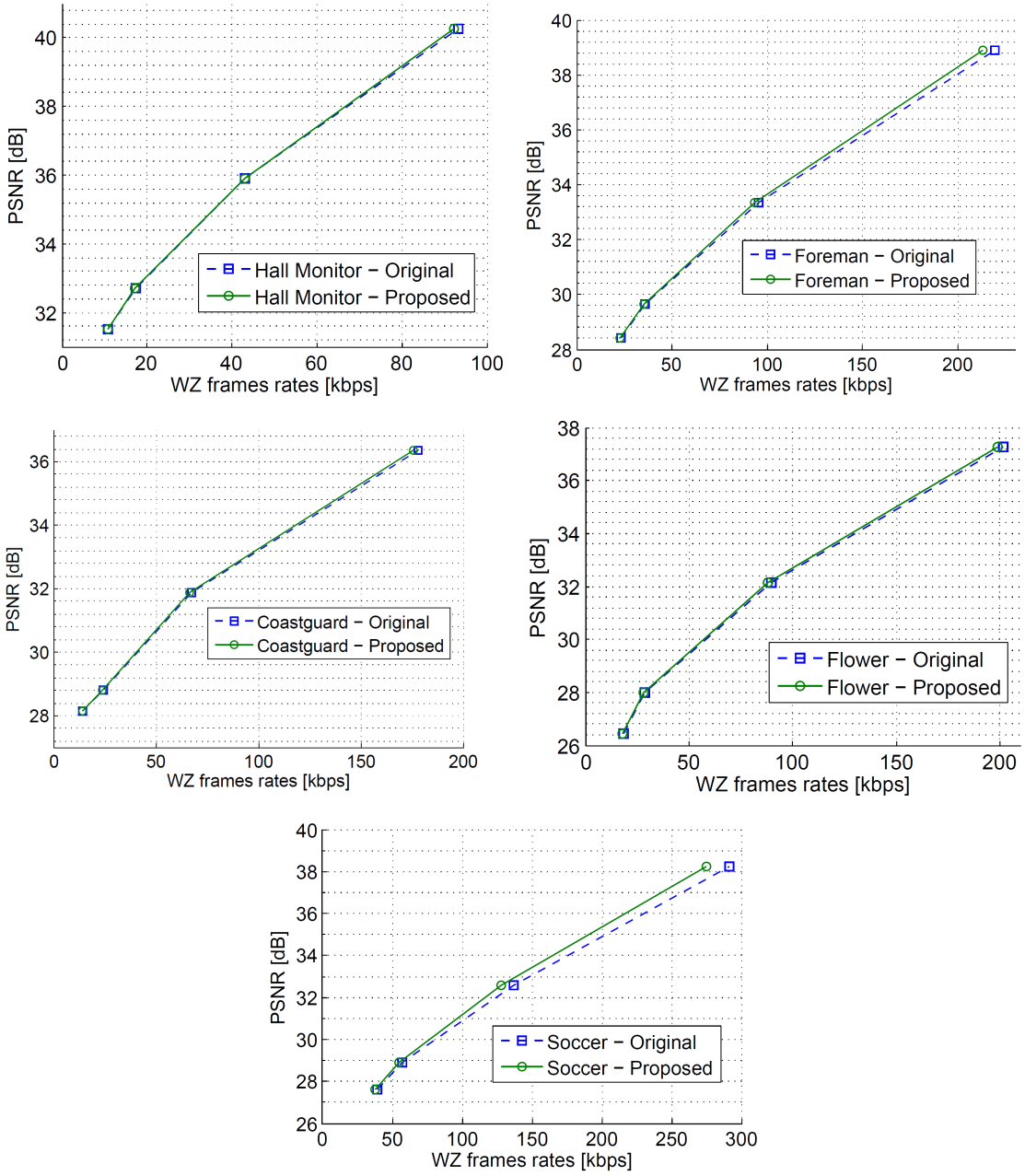

Figure 5.3: The proposed decoder that exploits the non-uniformity, while assessing the type of channel, needs less rate than the standard one to render the videos at the same PSNR.

if the true correlation model between the WZ bit planes and the side-information is *predictive* then the achievable rate remains that of uniform bit planes.

5.2 Gilbert-Elliott source model for DVC

The expected rate gain for the distributed coding of Gilbert-Elliott (GE) sources has been presented in Section 3.2 for synthetic sources. Here, we model the WZ bit planes as GE sources which parameters have to be estimated on-line. First, we investigate

Sequence	Rate gain (kbps)	Rate gain (%)
Hall Monitor	0.94	1
Foreman	5.88	2.68
Coastguard	2.04	1.15
Flower	2.97	1.47
Soccer	16.57	5.7

Table 5.2: Rate gain for the five video sequences, when the correlation is assessed *a posteriori* by the decoder.

the accuracy of the modeling, and we describe how to use the joint estimation-decoding EM algorithm described in Section 4.2 in the DISCOVER codec. Finally, we show some results for the coding of video sequences that prove the rate gain obtained with the GE source modeling, which therefore reveals to be a better model for the WZ bit planes than uniform or non-uniform sources models.

5.2.1 Accuracy of the GE source modeling

The question we deal with here is whether the bit planes are better approximated by the proposed GE source model or by a the non-uniform source model. To that end, we investigate the distribution of the bursts of 1's, i.e. the number of consecutive 1's, in the WZ bit planes.

First, consider a binary source X without memory. It can be modeled as a Bernoulli process with parameter $p_X = \mathbb{P}(X = 1)$. In this case, the burst length distribution, $(\mathcal{P}_k)_{k\geq 1}$, is defined as the probability of having a sequence with k consecutive 1's, given that the sequence starts with a 0 (i.e. $0\underbrace{1\ldots1}_{k}0$). For the Bernoulli process,

$$\mathcal{P}_k = (1 - p_X)p_X^{k-1} \tag{5.3}$$

Therefore, for Bernoulli sequences, the log-scale burst distribution $\log(\mathcal{P}_k)$ is linear in the burst length.

Now, consider a GE source. Let $\Pi = \frac{1}{t_{\mathbf{10}}+t_{\mathbf{01}}}\left[t_{\mathbf{10}}\ t_{\mathbf{01}}\right]$ be the steady-state distribution of the source. Let $B = \begin{pmatrix} p_{\mathbf{0}} & 0 \\ 0 & p_{\mathbf{1}} \end{pmatrix}$ be the matrix which diagonal values are the Bernoulli parameters. Let $P = \begin{pmatrix} (1-t_{\mathbf{01}}) & t_{\mathbf{01}} \\ t_{\mathbf{10}} & (1-t_{\mathbf{10}}) \end{pmatrix}$ be the state transition probability matrix. Let $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ be the identity matrix. We also introduce the following notation $C = \begin{pmatrix} 1-p_{\mathbf{0}} \\ 1-p_{\mathbf{1}} \end{pmatrix}$ and $\mathbb{I} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. The burst length distribution is thus given by:

$$\mathcal{P}_k = \frac{\Pi\,(I - B)\,(PB)^k\,C}{\Pi\,(I - B)\,(PB)^2\mathbb{I}} \tag{5.4}$$

Let us consider the bit plane sequences obtained within the DISCOVER codec for the sequence *Soccer*. For those sequences, we plot the *empirical* burst length distributions and the *theoretical* ones. The *empirical* distribution is obtained by

counting the occurrence of each burst length directly from the bit plane. Given the GE parameters estimated from a given bit plane, the *theoretical* burst length distribution is computed from Equation (5.4). Fig. 5.4 shows the comparison of the empirical distribution of a given bit plane (bold line) with two theoretical distributions (dot lines): one obtained by assuming that the bit plane has no memory, from Equation (5.3), and the other is obtained from Equation (5.4) by taking into account the memory. Interestingly, the empirical distribution of the bit plane matches well with the memory-aware theoretical distribution.

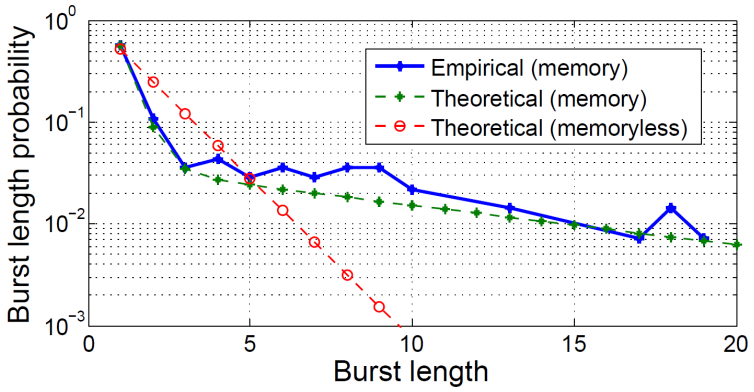

Figure 5.4: Distribution of the bursts of 1's for a selected bit plane of the video sequence *Soccer*, at the highest PSNR.

The plots in Fig. 5.4 only prove that the GE model is accurate for this particular bit plane that is analyzed. To quantify the memory in a broader number of bit planes, we look at the behavior of the parameter θ_X in the 100 first bit planes of the *five* video sequences. More precisely, for each bit plane, we observe $t_{\mathbf{10}}, t_{\mathbf{01}}$, and the ratio $\frac{p_1}{p_0}$. The presence of memory in the bit planes being conditioned by the combination of the three following criteria:

$$(a)t_{\mathbf{10}} \ll 0.5, \ \textit{and} \ (b)t_{\mathbf{01}} \ll 0.5, \ \textit{and} \ (c)\frac{p_{\mathbf{1}}}{p_{\mathbf{0}}} \gg 1 \tag{5.5}$$

The criteria (a) and (b) imposing low transition probabilities mean that the memory is *persistent*. The criterion (c) imposing a high ratio of the two Bernoulli parameters means that the two states are clearly distinguishable; otherwise, the two states are similar and the EM algorithm is not able to differentiate them.

The estimated transition parameters are shown in Fig. 5.5 for the *five* video sequences *Hall Monitor*, *Foreman*, *Coastguard*, *Flower*, and *Soccer*. Note that the transition probabilities can be very low, mainly for the sequence *Soccer*; this does account for a huge persistence of the states of the bit planes. The ratio of the estimated Bernoulli parameters are shown in Fig. 5.6 for the same video sequences, and the same 100 first bit planes. Here, we note that the ratio can be greater than 5 in some of the bit planes.

The combination of the low transition probabilities (Fig. 5.5) and the huge Bernoulli parameters ratio(Fig. 5.6) accounts for the presence of persistent memory in the WZ bit planes.

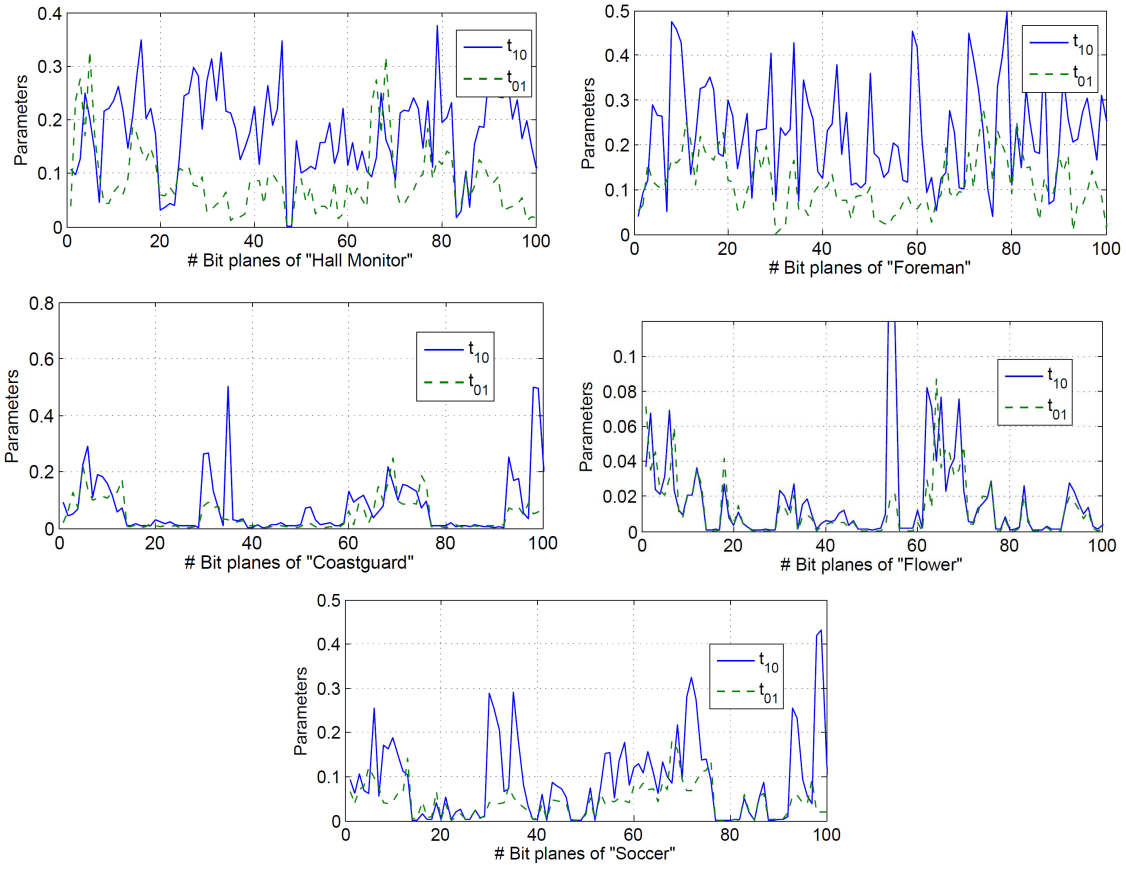

Figure 5.5: The estimated transition parameters from the 100 first bit planes from the five video sequences *Hall monitor, Foreman, Coastguard, Flower*, and *Soccer*.

5.2.2 Exploitation of the bit planes memory and the Laplacian correlation channel

For use with the EM algorithm presented in Section 4.2, one must take into account the fact that the intrinsic LLR is dependent on the nature of the correlation channel, and is computed as:

$$\log\left(\frac{\mathbb{P}\left(X_{k,n}^{b}=1|y_{k,n},x_{k,n}^{1},\ldots,x_{k,n}^{b-1}\right)}{\mathbb{P}\left(X_{k,n}^{b}=0|y_{k,n},x_{k,n}^{1},\ldots,x_{k,n}^{b-1}\right)}\right)$$

$$=\begin{cases}\log\left(\dfrac{\int\limits_{x\in Q(1)}\frac{\alpha_k}{2}e^{-\alpha_k|y_{k,n}-x|}dx}{\int\limits_{x\in Q(0)}\frac{\alpha_k}{2}e^{-\alpha_k|y_{k,n}-x|}dx}\right), & if\ the\ channel\ is\ predictive\\ \log\left(\dfrac{\int\limits_{x\in Q(1)}\frac{\alpha_k}{2}e^{-\alpha_k|y_{k,n}-x|}dx}{\int\limits_{x\in Q(0)}\frac{\alpha_k}{2}e^{-\alpha_k|y_{k,n}-x|}dx}\right)+\log\left(\dfrac{1-p_{X_n}^{l}}{p_{X_n}^{l}}\right), & if\ the\ channel\ is\ additive\end{cases}$$

where p_{Xn}^{l} is the instantaneous Bernoulli parameter estimated at iteration l of the EM algorithm. It is updated thanks to the update rule given in Equation (4.13).

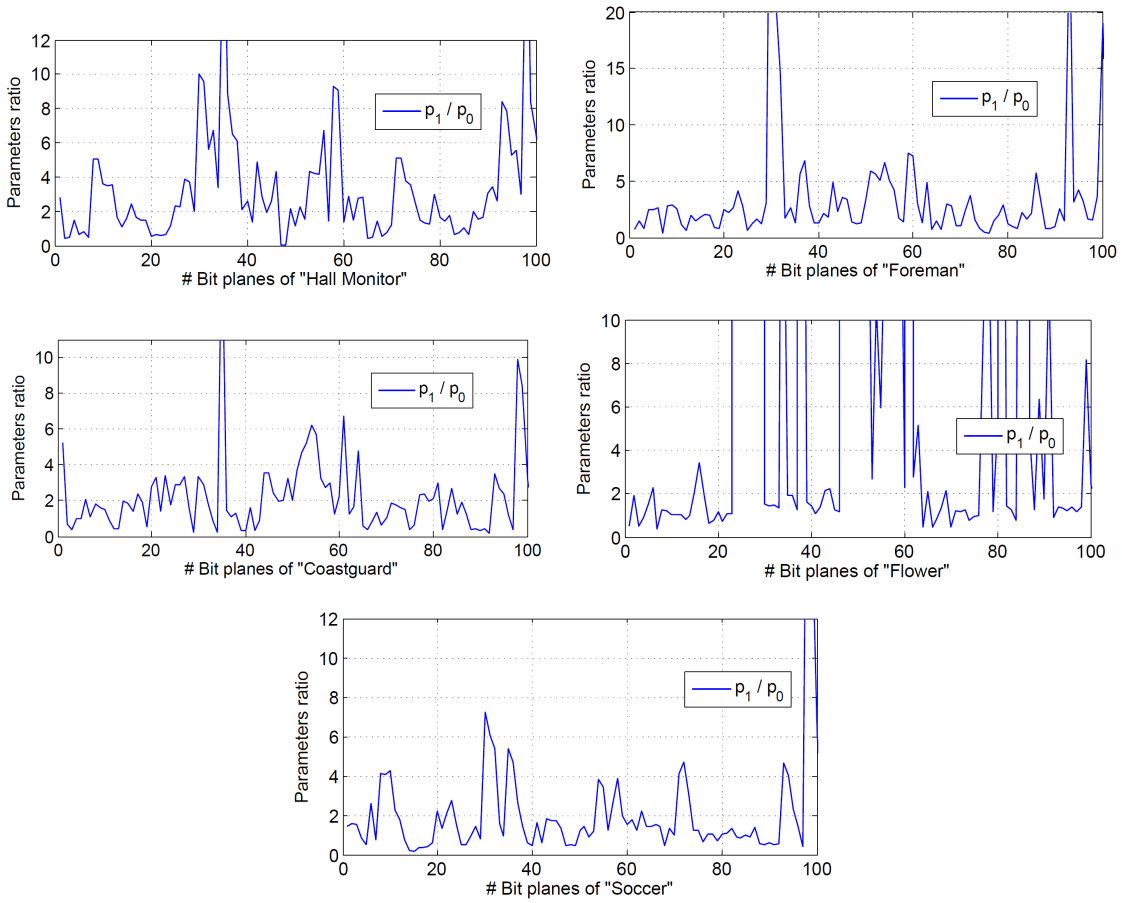

Figure 5.6: The ratio of the estimated Bernoulli parameters from the five video sequences *Hall monitor, Foreman, Coastguard, Flower*, and *Soccer*.

Practically, for the message-passing algorithm in Section 4.2.5, the channel-to-variable message, B_n in the Equation (4.14), is replaced by

$$\forall n \in [1, N], B_n = \log\left(\frac{1 - p^b_{k,n}}{p^b_{k,n}}\right) \tag{5.6}$$

The definitions of the other messages for the LDPC decoding are not modified.

The minimum rate needed for the rate-adaptive decoding is not assessed here, since the effects of the memory on the original minimum rate in Equation (2.6) has not been solved yet. For the decoding, we use the minimum rate computed for non-uniform sources in Section 5.1.2.2, which is a lower bound on the minimum rate needed for memory sources.

5.2.3 Implementation and experimental results

In order to assess the performance achieved by the proposed Slepian-Wolf decoder that exploits the memory in the bit planes, we integrate the LDPC-based estimation-decoding EM algorithm in the DISCOVER codec. All the bit planes do not have memory: some can be *i.i.d.* binary sources. But this model is a particular case of the

GE model, so the decoder adapts by itself to any of the source models. The channel modeling the correlation between the WZ bit planes and the SI can be *additive* or *predictive*, and we cannot know the true channel model *a priori*, since we do not have access to the true WZ data. Therefore, the BSC model has to be estimated dynamically on-line as for the non-uniform source modeling. In order to prove and to exploit the existence of the *predictive* channel, we first carry out the decoding assuming that the channel is only *additive*.

5.2.3.1 The channel is assumed to be only additive

We show in Fig. 5.7 the performance of the modified decoder if the noise between the WZ bit planes and the SI is assumed to be *additive*. For the sequence *Soccer*, the rate gain is almost 10% at the highest PSNR, with respect to the standard SW decoder of the DISCOVER codec. However, we see that there is not always a rate gain for the exploitation of the GE model for the bit planes; for the sequence *Foreman*, the standard and the proposed decoder have the same performance; for the sequence *Hall Monitor*, the performance of the proposed decoder is degraded compared to that of the standard decoder.

The rate gain is detailed in Table 5.3 when the correlation is assumed to be only additive and the video PSNR is the highest.

Sequence	Rate gain (kbps)	Rate gain (%)
Hall Monitor	−1.84	−1.98
Foreman	−0.87	−0.4
Coastguard	−2.79	−1.57
Flower	3.15	1.56
Soccer	26.72	9.17

Table 5.3: Rate gain for the five video sequences, when the correlation is assumed to be additive only.

From these figures, we can note that modeling the bit planes as GE sources, instead of non-uniform sources, makes the mismatched decoder less vulnerable to rate loss, since the loss is small for all the video sequences with respect to the loss observed in Table 5.1 for non-uniform source modeling.

5.2.3.2 The channel is unknown and assessed by the decoder

Now, we do not make any *a priori* guess on the correlation channel, and we estimate it on the fly at the decoder. To that end, for each additional syndrome request made by the decoder, the decoding is first performed with the *additive* channel assumption; if this first decoding fails, (see the failure criteria in Section 4.2.6), another decoding is carried out, assuming that the channel is *predictive*. The results presented in Fig. 5.8 are obtained for all the WZ frames of the video sequences *Hall Monitor*, *Foreman* and *Soccer*.

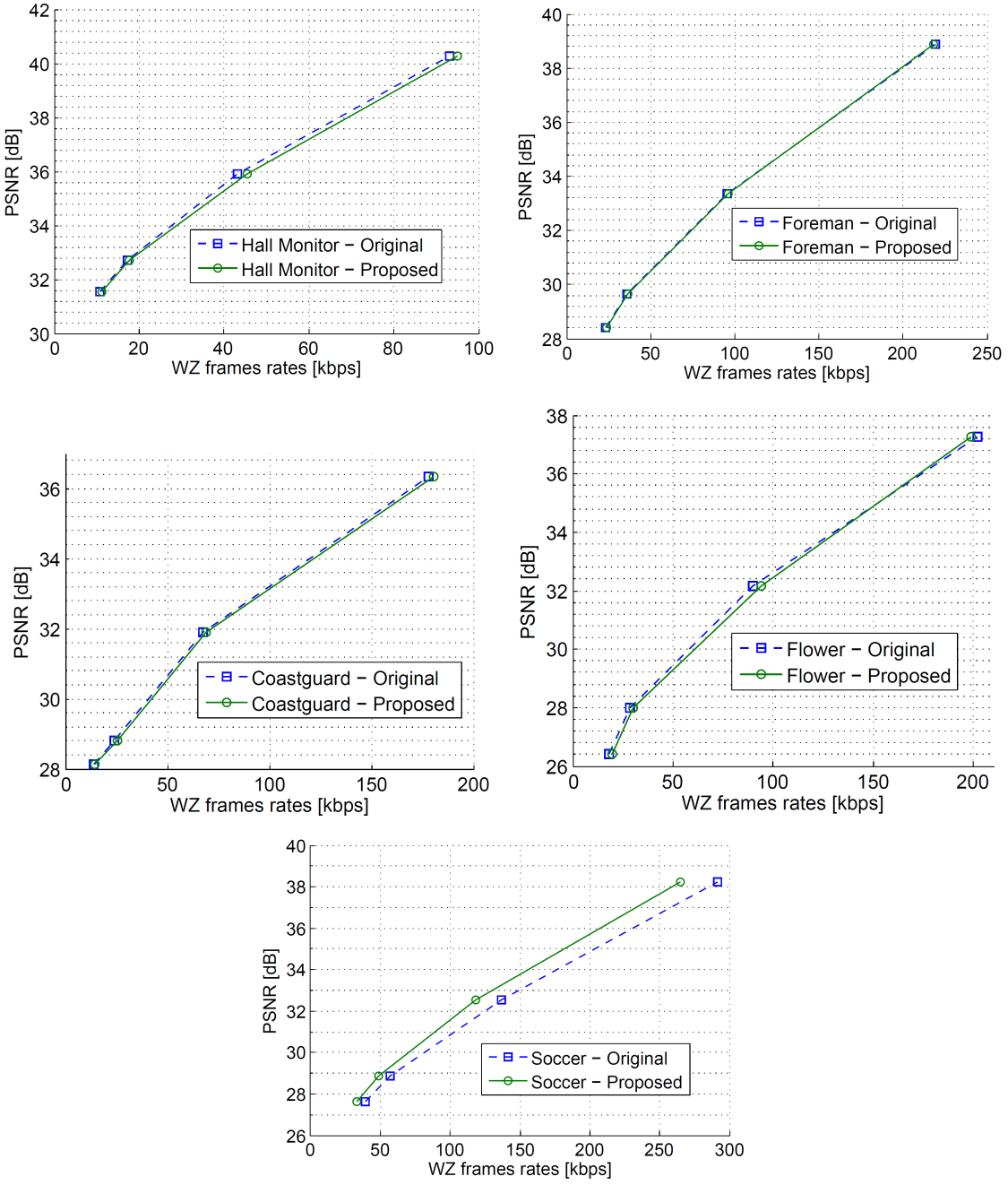

Figure 5.7: Behaviour of the proposed DVC decoder when the Laplacian channel is assumed to be always additive. Exploiting the memory that is present in the bit planes does not always improve the performance of the codec.

While the model remains relatively simple (only two states are needed to model an infinite memory in the bit planes), the gain, in terms of decreasing the rate allocated to the WZ frames, is significant. The decrease of rate for the sequence *Soccer* is $29.55kbps$ (10.14%) at the highest PSNR. This proves that it is worth taking into account the memory estimated from the bit planes, and that the additive channel is not sufficient to model the correlation between the WZ bit planes and

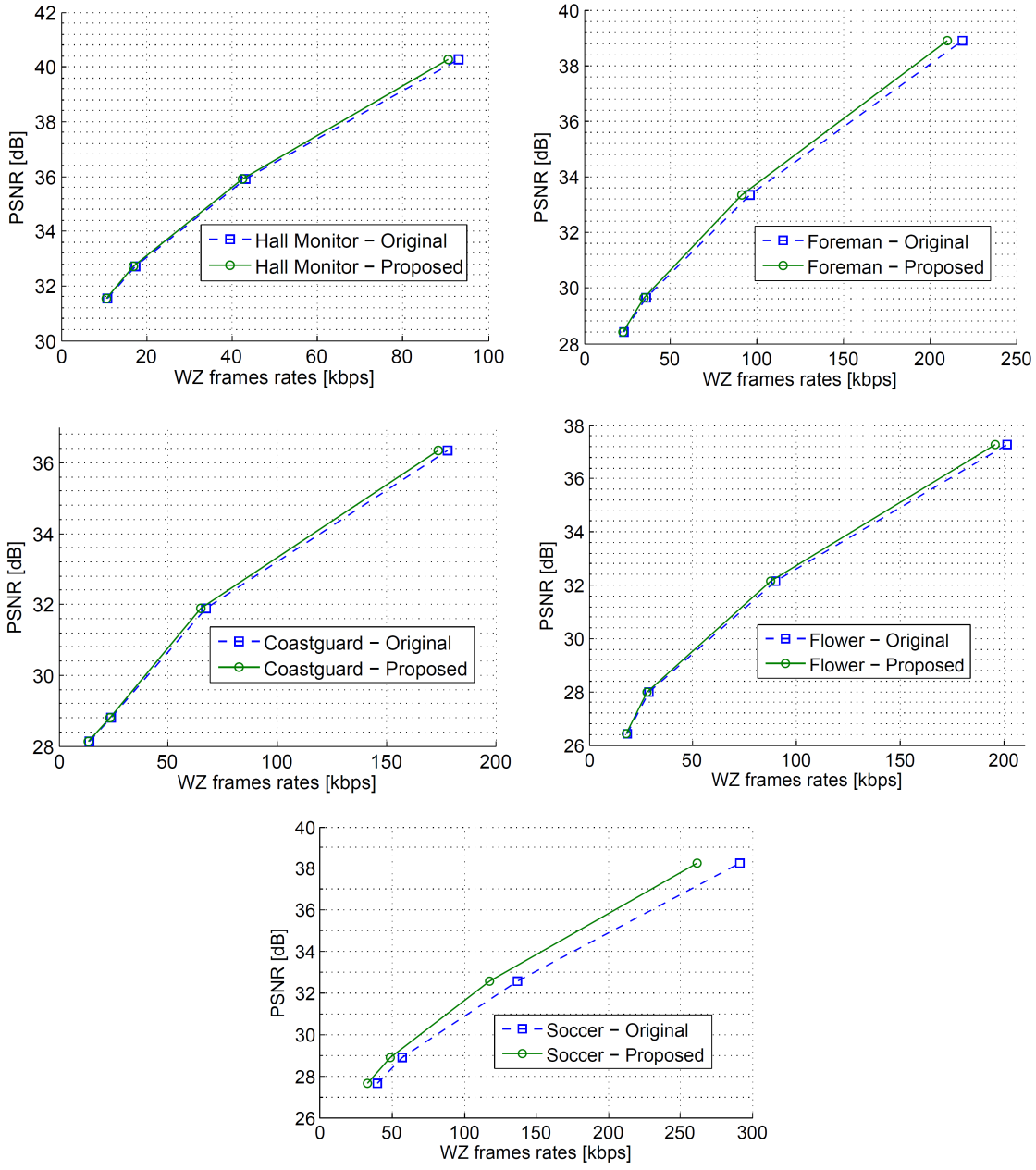

Figure 5.8: The decoder that exploits the bit planes memory needs less rate to render the videos at the same PSNR.

the SI generated by the DVC codec. The rate gain is given in Table 5.4 for the *five* video sequences at the highest PSNR.

These figures can be compared with those for non-uniform source modeling in Table 5.2. Therefore, we can quantify the rate gain obtain from the GE source modeling, with respect to the non-uniform source modeling. More precisely, the GE source modeling is 2.7 times better for the coding of the sequence *Hall Monitor*, 1.49 times better for the coding of *Foreman*, 2.1 times better for *Coastguard*, 2 times better for *Flower*, and it is 1.78 times better for the sequence *Soccer*.

Sequence	Rate gain (kbps)	Rate gain (%)
Hall Monitor	2.54	2.73
Foreman	8.76	4
Coastguard	4.28	2.41
Flower	5.96	2.95
Soccer	29.55	10.14

Table 5.4: Rate gain for the five video sequences, when the correlation is assessed *a posteriori* by the decoder.

5.3 Markov chain modeling: particular case of Hidden Markov modeling

The *Markov source model* is the simplest representation of the memory that lies in the binary bit planes. The Markov source is a particular instance of the two-state hidden Markov source, since it corresponds to the case where $p_{\mathbf{0}} = 0$ and $p_{\mathbf{1}} = 1$, regardless of the state transition probabilities. This implies that the states are directly observable from the source realization itself; *i.e.* $\forall n \in [1, N], \sigma_n = \mathbf{1}x_n + \mathbf{0}(1 - x_n)$. The transition parameters can also be directly computed from the observed source realization.

We modify the estimation-decoding EM algorithm to estimate the states and the transition probabilities for this Markov modeling. Then, we place the modified SW decoder in the DISCOVER codec. The results that we obtained show that there is practically no rate gain for the coding of the WZ bit planes, with respect to the standard DISCOVER codec, even if the channel model (additive or predictive) is dynamically assessed on-line. More precisely the rate improvement is only $0.27kbps$ for the sequence *Hall Monitor* at the highest PSNR; it is $0.18kbps$ for *Foreman* and $1.15kbps$ for *Soccer*. It is worth noting that the non-uniform source model (Section 3.1.1) brings more improvement than the Markov one, in terms of the PSNR versus rate performance of the DISCOVER codec. This might be contra-intuitive, but note also for two correlated Markov sources X and Y, the states of X and Y do not correspond (more precisely, Lemma 3.1 do not stand for Markov sources); we can say that the knowledge of the states of Y is misleading for the decoding of X.

Therefore, the Markov source model is not suited to model the memory that lies in the WZ bit planes generated by the DVC codec.

5.4 Performance comparison with State-Of-The-Art codecs

Throughout this Chapter, we have presented the improvement that we brought to the distributed video codec DISCOVER, by comparing the outcome of our contributions to the last version known of the codec (implemented in 2007). Here, we compare the actual performance with two codecs that are known in the literature,

namely H264/AVC in the Intra mode and H264/AVC in the Inter mode without motion estimation (to make a fair comparison with the low-complexity encoder of DISCOVER). The results are presented in Fig. 5.9 for the five video sequences *Hall monitor, Foreman, Coastguard, Flower*, and *Soccer*.

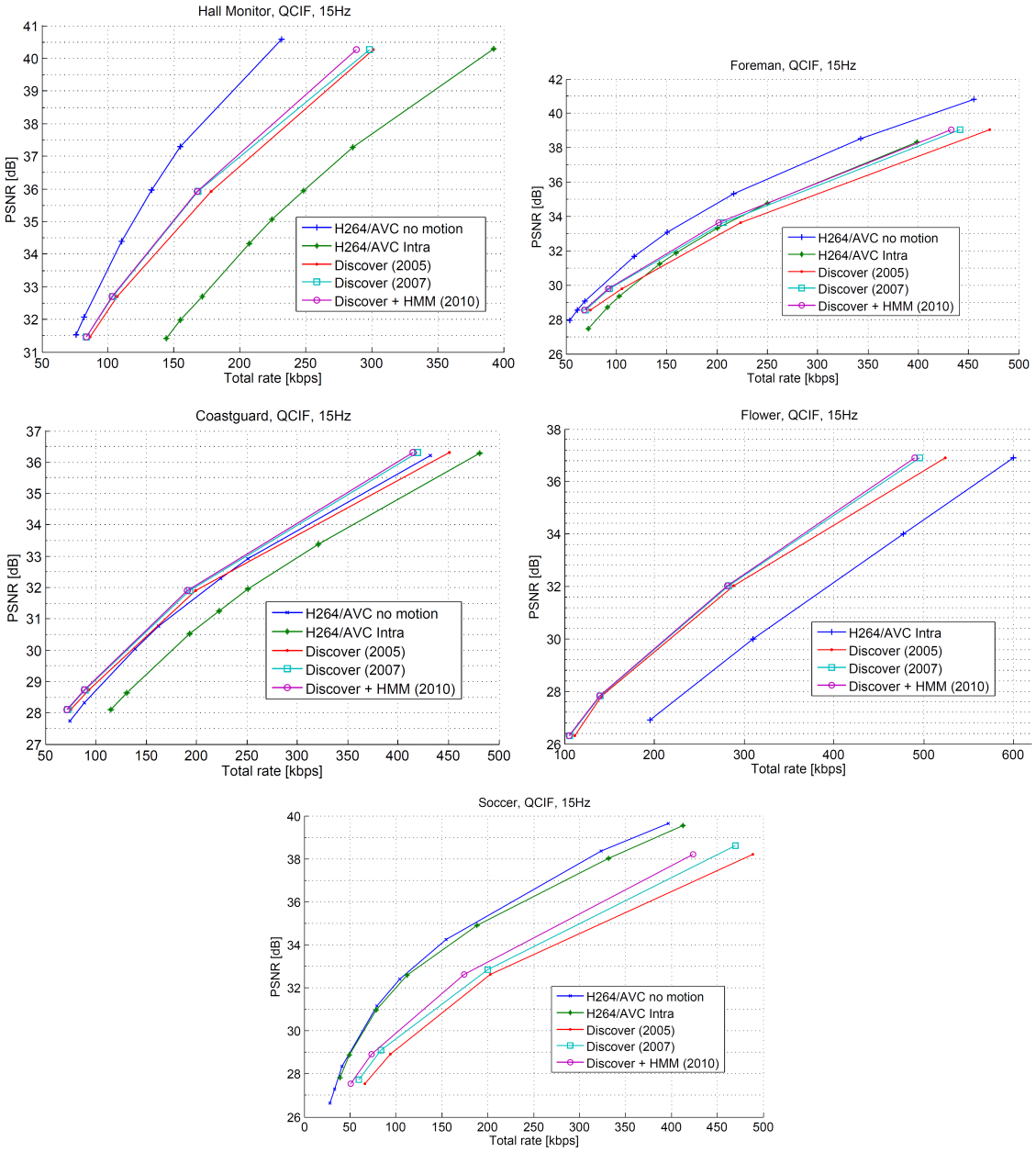

Figure 5.9: Performance comparison of three versions of DISCOVER with State-Of-The-Art codecs for the five video sequences.

Three versions of DISCOVER are presented:

- The first version, that was implemented at the launch of the European Project, in 2005;
- The last known version of the codec, implemented in 2007, before we brought the improvement described in this Thesis;

- The actual version of the codec, with the modeling of the bit planes as Hidden Markov Models (HMM, 2010).

Our version of the DISCOVER codec performs always better than the two previous versions, thanks to the HMM modeling of the bit planes. When we compare to the performance of H264 codec, the results are mitigated since they depend on the sequence that is analyzed. As can be seen on the pictures in Fig. 5.9, the performance of the two versions of H264 differ very much from video sequence to video sequence: the DISCOVER codec outperforms the Intra mode for the sequences *Hall monitor* and *Coastguard*, but it is far behind for the sequence *Soccer*, and it has comparable results for *Foreman*. The H264 in Inter mode without motion estimation is far better than the DISCOVER codec for the sequences *Hall Monitor, Foreman* and *Soccer*, but it performs worse for the sequence *Flower*, and it has comparable results for *Coastguard*.

5.5 Conclusion

The models and tools introduced in the two previous Chapters have been used so as to improve the rate-distortion performance of the DISCOVER codec. The idea has been to analyze the binary versions of the WZ frames (the so-called WZ bit planes) to exhibit a more pertinent model than the usual uniform model assumed by the standard SW decoder. At first look, a better model is the non-uniform Bernoulli source model, which remains memoryless but insures a rate gain that increases with the non-uniformity. The resulting rate gain is up to 5.7% for the coding of the WZ frames without any loss of quality, provided that the correlation model (*additive* or *predictive*) is not mismatched. If we consider a memory source model, Gilbert-Elliott sources in our case, the rate gain is even greater and allows a rate gain up to 10.14% for the coding of the WZ frames, again provided that the correlation model is well assessed.

Conclusion and perspectives

Conclusion

This thesis presented the work on Distributed Source Coding (DSC) and Distributed Video Coding (DVC) that I carried out during my PhD. Three main contributions were developed:

1. The investigation of pertinent models for distributed coding of binary sources, with the aim to apply these models to DVC. Two source models were exhibited, namely non-uniform sources and hidden Markov sources. If the former is a special case of hidden Markov sources, the two models imply very different achievable SW bounds, when compared to the traditional distributed coding of uniform sources. Both models are well suited for the bit planes generated by a DVC codec and they let the codec be improved consequently. Whatsoever, rate gain for DVC is not always observable when applying both models. This is due to the underlying correlation channel that is not symmetric with the correlated sources when they are not uniform. Therefore, we had to take into account that asymmetry by distinguishing between the additive correlation channel (which is actually the traditional channel model) and the predictive correlation channel. We emphasized on the drastically different achievable bounds for both channel models, and we designed practical syndrome-based Turbo and LDPC codes that are able to account for the distributions of the sources and adapt to the correlation model as well. When the correlation is modeled as a BSC, we proposed an estimation algorithm of its parameter, based on the EM algorithm. The most important part of our contribution to BSC parameter estimation is the initialization of the EM; more precisely, we showed that there is a direct one-to-one correspondence between the Bernoulli parameter of the BSC and the Bernoulli parameter of its syndrome. An ML algorithm was thus developed to approach the parameter before the decoding.

2. The development of several practical DSC tool, based on syndrome-based Turbo and LDPC codes, that achieve the asymmetric SW bounds as well as the non-asymmetric one. For asymmetric coding, we presented DSC codes that are able to take into account the source distribution and the model of the correlation channel. The work on LDPC codes is presented, but the same results could also be exhibited for Turbo codes. The EM algorithm is used for the parameter estimation. When combined with iterative decoding algorithms, the EM becomes an efficient joint estimation-decoding algorithm. For non-asymmetric coding, we first presented a novel DSC code, based on linear block codes, that is able to reach the non-asymmetric SW bound for a given correlation and for uniform sources; when the correlation varies, accumulating the syndrome yields a rate-adaptive DSC code. Nonetheless, this system is subject to the error propagation phenomenon. Therefore, we investigated the conditions under which the errors do not propagate, and we implemented practical DSC codes to avoid the error propagation. When the sources are uniform, the compression problem is symmetric with the sources. However, when dealing with non-uniform and memory sources, the channel can be additive or predictive, and it must carefully be taken into account in the decoding.

3. The integration of these models and tools to a DVC codec, which SW part is based on rate-adaptive LDPC codes. The SW decoder of the DISCOVER codec was substituted with the EM algorithm to exploit the non-uniformity of the generated bit planes, but also their memory when they are modeled as hidden Markov sources. The source parameters are initialized to that of the side-information bit planes. We empirically showed that the distribution of the bit planes can reasonably be modeled as non-uniform sources as well as hidden Markov sources. We also presented experimental results that prove that the channel between the source video bit stream and the side-information can be additive as well as predictive, for both non-uniform sources and hidden Markov. As the correlation model cannot be estimated prior to decoding, the SW decoding must test both possibilities and choose the one that provides the best rate-distortion performance.

If the hidden Markov source modeling is better for DVC, in terms of improving its rate-distortion performance, it is also the most expensive solution, in terms of increasing the decoder's complexity. Therefore, a clever trade-off must be found; if the decoder is powerful enough, then the hidden Markov model could be used; however if the decoder must render the video at a high frame rate then the non-uniform source model is far acceptable, even if real time playing remains unreachable.

Perspectives

Non-asymmetric SW coding of GE sources

We can extend the results exhibited in Section 4.3.3 for non-asymmetric SW coding of non-uniform sources. The problem of channel asymmetry is similar to the case of non-uniform sources. Suppose we want to compress two GE sources X and Y *s.t.* $X \sim \mathcal{B}(\theta_X)$, and $Y = X \oplus Z$, where Z is some correlation noise that is modeled as a BSC of parameter p. Therefore, the correlation channel is additive for the coding of X and it is predictive for the coding of Y, meaning that the theoretical achievable bounds for each source are not the same: exploiting the memory in X will increase its conditional entropy-rate, *i.e.* $H(\mathcal{X}|\mathcal{Y}) = H(Z) - [H(\mathcal{Y}) - H(\mathcal{X})]$, while the memory in Y is not exploitable, *i.e.* $H(\mathcal{Y}|\mathcal{X}) = H(Z)$.

The decoding has also to take into account that the states of the source are not known, so a modified EM algorithm has to be implemented so as to jointly exploit the memories in X and Y. Actually, even if the memory does not bring any rate improvement for the decoding of Y, the states of X and Y are still strongly correlated. Therefore, the basis of the joint factor graph of the problem is the same as in Fig. 4.20, but the state nodes have also to be taken into account, as in the factor graph in Fig. 4.6. Thus, the message passing algorithm is modified so as to make an estimation of the states (and the parameters of the sources, if they are unknown) at each iteration. The knowledge of the systematic source bits is an asset to the decoding since reliable estimation of the sources parameters can be done, prior to the decoding itself. The challenge here is to estimate the states in the unknown parts x_{K+1}^N and y_{K+1}^N since no side-information is available for both sources.

Non-asymmetric and rate-adaptive SW coding of non-uniform and memory sources

The idea is to combine the results presented in Section 4.3.3 for efficient non-asymmetric SW coding of non-uniform sources, and in Section 4.3.1 for efficient rate-adaptive SW coding of uniform sources, to create a unique codec that is able to reach any point in the SW rate region, for the case where the sources are non-uniform or have memory. The main issue that we have noted is that, in the scheme proposed in Section 4.3.1, both sources send the same amount of syndrome information for any code rate under consideration; and the same issue is observed in Section 4.3.3, since given the code chosen for the non-asymmetric coding, the two sources also send the same amount of syndrome bits. Meanwhile, the conditional entropies of the correlated sources are not equal if they are not uniform. So the proposed schemes need to be adapted to the coding of sources that are not uniform.

Non-binary channel parameter estimation

We want to generalize the scheme adopted in Section 3.4, for the BSC parameter estimation, to the estimation of the parameters of non-binary channels. For example, we place ourselves in the case of a binary input Gaussian channel, and we want to estimate the Gaussian parameters when we only observe the syndrome of the binary source X and the continuous-valued side-information Y, given the parity-check matrix of the code. In this case, we can compute a continuous-valued "syndrome" of the realization $\mathbf{y}$ by a multiplication with $\mathbf{H}$. Then, combining the syndromes of the realization $\mathbf{x}$ and that of $\mathbf{y}$, we should find a function of the Gaussian parameter. This is based on the fact that the result of the sum of independent Gaussian random variables is also a Gaussian random variable.

On-line assessment of the BSC model (additive/predictive)

In this Thesis, we showed the difference between the additive BSC and the predictive BSC, in terms of exhibiting achievable SW bounds for the correlated sources and in terms of demonstrating how to adapt the DSC codes to reach these bounds, knowing the model. However, given the side-information $Y \sim \mathcal{B}(p_Y)$ and the syndrome of $X \sim \mathcal{B}(p_X)$, we have not shown how to guess the BSC model. This on-line assessment would avoid to decode the data two times for both non-uniform and GE source modeling, for distributed video compression. Our idea is to use the BSC parameter estimator presented in Section 3.4, to guess p_X and p_Y and to deduce if the model is additive or predictive (if p_X and p_Y are both different from 0.5). Actually, a simple test is the sort p_Y with respect to p_X and decide: if p_Y is closer to 0.5 than p_X, then $H(p_Y) \geq H(p_X)$, then the model should be additive; and it should be predictive otherwise. However, such a method would not work if X and Y are drawn according to a distribution different from the non-uniform one.

Robust video coding using ideas from WZ coding

When video sequences are transmitted through an error-prone transmission channel, for example the Internet, the video quality is degraded because of packets losses or other errors. The idea here is to use the results from Distributed Source Coding (DSC) to protect the video frames against the degradations. Traditionally, the scheme for video compression and transmission is composed of an inner channel codec and an outer video codec. At the encoder, the video is first compressed, then some redundant information is added to the data flow to fight against channel transmission errors. At the decoder, the data flow is first channel decoded, and then the video is decompressed. This traditional scheme is subject to *drift* effects and to *Cliff* effects, when the channel error rate is above the channel code's correction capability.

Using DSC, one can transmit the compressed video over the channel, along with a coarser description that takes the form of a syndrome. The idea is to use the received erroneous version of the decompressed video as side-information for WZ decoding exploiting the additional information from the syndrome. This novel scheme is called "Systematic Lossy Error Protection" (SLEP), and it is presented in the literature [RAG04, RBG08] by Rane *et. al.* Our aim is to apply the models that we presented in Chapter 3, and the estimation-decoding tools presented in Chapter 4 to improve the decoding performance of the SW decoder.

Part III

Annexes

Appendix A

Row echelon form of a binary matrix for efficient non-asymmetric syndrome-based SW coding

This chapter formally describes the theory behind the search for a subset of linearly independent columns of a binary matrix. The algorithm for such a search is also properly exhibited.

Let $\mathbf{H}$ be a binary matrix of size $M \times N$ s.t. $M \leq N$. The rank of $\mathbf{H}$, noted $Rank(\mathbf{H})$, is the number of *linearly independent* column vectors composing it (also called *free* columns). $Rank(\mathbf{H}) \leq M$. Formally, the free columns form a basis for the subspace generated by the columns of $\mathbf{H}$.

A.1 Prerequisite

The algorithm consists into putting the binary matrix in an *echelon form*, by applying only elementary *row* operations. This treatment does not modify the rank of the matrix and, more importantly for the application, it does not change the position of the free columns (see Theorem A.1). The resulting Row Echelon Form (REF) of the binary matrix satisfies the following simple rule:

- $\oplus$ $\forall m \in [1, M]$ The first non-zero element of row m, a "1" at the n-th position in the row, is the last non-zero element of the column n. This "1" is called the *pivot* of the row m and the column n.

Note that in the *Gauss pivot* algorithm, the pivot is the only non-zero element of the row *and* the column, and the pivots are echeloned so as to form the identity matrix. The REF algorithm stops before arriving to the Gauss-pivot form: in each row, we find the *pivot* as the first 1 in the row, and we zero only the elements in the corresponding column that have a larger label than the pivot; no permutation between the columns is performed.

Note also that zeroing the elements in the column that come before the pivot brings to the Reduced Row Echelon Form (RREF) of the matrix $\mathbf{H}$.

Example of row echelon form of a matrix:
In the following, $\mathbf{H}$ is put to its REF $\mathbf{H}'$:

$$H = \begin{bmatrix} 0&1&1&0&0&1&1&1&0&0&0&0 \\ 1&0&1&0&1&1&1&0&1&0&0&0 \\ 1&1&0&0&1&1&0&0&1&1&1&0 \\ 1&1&1&1&0&1&0&1&1&1&0&1 \\ 0&0&0&0&1&0&0&1&1&1&0&1 \\ 0&1&0&1&0&0&0&1&0&1&1&1 \\ 1&1&0&1&0&1&0&0&1&1&1&0 \\ 1&0&1&1&1&0&1&0&1&1&1&1 \end{bmatrix} \Rightarrow H' = \begin{bmatrix} 0&\mathbf{1}&1&0&0&1&1&1&0&0&0&0 \\ \mathbf{1}&0&1&0&1&1&1&0&1&0&0&0 \\ 0&0&0&0&0&\mathbf{1}&0&1&0&1&1&0 \\ 0&0&\mathbf{1}&1&1&0&0&1&0&0&1&1 \\ 0&0&0&0&\mathbf{1}&0&0&1&1&1&0&1 \\ 0&0&0&0&0&0&\mathbf{1}&1&1&1&1&1 \\ 0&0&0&\mathbf{1}&0&0&0&1&1&1&0&1 \\ 0&0&0&0&0&0&0&0&\mathbf{1}&1&0&0 \end{bmatrix}$$

The *pivots* are put in bold face in $\mathbf{H}'$.

In the sequel, we call *echeloned column* a column containing a pivot.

Lemma A.1. *When* $\mathbf{H}$ *is put under the row echelon form, the echeloned columns are linearly independent.*

Proof. Suppose that the matrix $\mathbf{H}$ is put under its REF. Then, within a permutation of the columns, the echeloned columns can be ordered according to the increasing value of the label of the pivot in the column. The echeloned columns of the obtained matrix form an upper triangular square matrix; thus these columns are linearly independent. □

Definition A.1. *Two matrices are row-equivalent if they can be obtained from one another by applying only elementary row operations.*

Theorem A.1. *Let* $\mathbf{H}$ *and* $\mathbf{H}'$ *be two row-equivalent matrices. A subset of the columns of* $\mathbf{H}$ *is linearly independent if, and only if, the corresponding subset of columns of* $\mathbf{H}'$ *is also linearly independent.*

Proof. Consider these two following free columns of $\mathbf{H}$:

$$C_a = \begin{bmatrix} C_{a1} \\ C_{a2} \\ \dots \\ C_{aj} \\ \dots \\ C_{ak} \\ \dots \\ C_{am} \end{bmatrix}, \text{ and } C_b = \begin{bmatrix} C_{b1} \\ C_{b2} \\ \dots \\ C_{bj} \\ \dots \\ C_{bk} \\ \dots \\ C_{bm} \end{bmatrix}$$

Suppose that $\mathbf{H}'$ is obtained from the elementary operation on the row m of $\mathbf{H}$: $L_m \leftarrow L_m + \alpha L_k$. Then the corresponding columns of $\mathbf{H}'$ are:

$$C'_a = \begin{bmatrix} C_{a1} \\ C_{a2} \\ \dots \\ C'_{am} = C_{am} + \alpha C_{ak} \\ \dots \\ C_{ak} \\ \dots \\ C_{am} \end{bmatrix}, \; and \; C'_b = \begin{bmatrix} C_{b1} \\ C_{b2} \\ \dots \\ C'_{bm} = C_{bm} + \alpha C_{bk} \\ \dots \\ C_{bk} \\ \dots \\ C_{bm} \end{bmatrix}$$

Since the two columns C_a and C_b are free, $\forall \lambda, \mu, \lambda C_a + \lambda C_b = 0 \Rightarrow \lambda = 0, \mu = 0$; without loss of generality, suppose that:

$$\begin{cases} \lambda C_{am} + \mu C_{bm} = 0 \\ \lambda C_{ak} + \mu C_{bk} = 0 \end{cases} \Rightarrow \lambda = 0, \mu = 0$$

Now, consider the corresponding system in $\mathbf{H}'$:

$$\begin{cases} \lambda C'_{am} + \mu C'_{bm} = 0 \\ \lambda C_{ak} + \mu C_{bk} = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} \lambda(C_{am} + \alpha C_{ak}) + \mu(C_{bm} + \alpha C_{bk}) = 0 \\ \lambda C_{ak} + \mu C_{bk} = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} \lambda C_{am} + \mu C_{bm} + \alpha(\lambda C_{ak} + \mu C_{bk}) = 0 \\ \lambda C_{ak} + \mu C_{bk} = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} \lambda C_{am} + \mu C_{bm} = 0 \\ \lambda C_{ak} + \mu C_{bk} = 0 \end{cases} \Rightarrow \lambda = 0, \mu = 0$$

Thus the two corresponding columns in $\mathbf{H}'$ are also free.

All the elementary row operations are invertible. Then, the converse of Theorem A.1 is demonstrated using the same argument, by noting that $\mathbf{H}$ is obtained from $\mathbf{H}'$ by applying elementary row operations. Thus the linear independence of a subset of columns of the REF matrix $\mathbf{H}'$ also implies the linear independence of the corresponding columns of the original $\mathbf{H}$.

□

A.2 Statement of the REF algorithm

Given the results in the previous Section, the algorithm for the search of free columns in $\mathbf{H}$ is formulated as:

1. By applying only elementary *row* operations, reduce $\mathbf{H}$ under its row echelon form $\mathbf{H}'$;
2. Locate the columns of $\mathbf{H}'$ that contain the pivots, they are free columns of $\mathbf{H}'$;
3. The positions of the free columns of $\mathbf{H}$ are the same as those of $\mathbf{H}'$.

Note that these three steps are performed simultaneously in the practical implementation that we designed.

A.3 Optimizing the search

In the case where the matrix $\mathbf{H}$ is of rank M, let $\mathbf{B}$ denote the square matrix composed of a subset of free columns of $\mathbf{H}$. Now, note that the algorithm that we describe in the previous Section only shows the existence of one such $\mathbf{B}$. This comes from the fact that only one possibility is available for the choice of the pivot: it is the first "1" in the row. However, we can obtain *all* the subsets of M free columns of $\mathbf{H}$ by randomly choosing the pivots among all the non-zero elements of each row under consideration. Therefore, we can choose the "best" matrix $\mathbf{B}$ that fulfill arbitrary conditions. For example, we can choose the one that minimizes the weight of the inverse $\mathbf{B}^{-1}$ (for use with the non-asymmetric Slepian-Wolf setup in Sections 4.3, 4.3.2, and 4.3.3, to reduce the *error propagation* phenomenon).

Appendix B

Progressive edge growth algorithm for fast and efficient non-asymmetric SW coding

In this Annex, we present the Progressive Edge Growth (PEG) [HEA05] adapted to the construction of LDPC codes that fulfill the condition enumerated in Section 4.3.3.3 for efficient non-asymmetric coding of Bernoulli sources. Remind that the condition imposes that the matrix $\mathbf{H}$ contains an invertible square sub-matrix $\mathbf{B}$ that is triangular (given some permutation of its columns). However, the construction must not deteriorate the performance of the code, in the sense that the resulting matrix must respect the variable and check degree distributions found via density evolution [RSU01]. Therefore, we must modify the original PEG so as to obtain the best matrix. Our solution is to exploit the results presented in [JW05] for the construction of IRA codes. First, we review the original PEG, and then we deduce the algorithm adapted to our problem.

B.1 The original Progressive Edge Growth algorithm

B.1.1 Notation and definition

An (N, K) syndrome-based LDPC code can be interchangeably represented by an $(N-K) \times N$ *sparse matrix* $\mathbf{H} = (h_{ij})_{i\in[1,N-K],j\in[1,N]}$, or by a *Tanner graph* G with $(N-K)$ *check nodes* $\mathbf{s} = (s_i)_{i=1}^{N-K}$, representing the syndrome symbols, and N *variable nodes* $\mathbf{x} = (x_j)_{j=1}^{N}$, representing the source symbols. Check nodes are connected to variable nodes by *edges*, s.t. s_i is connected to x_j iff $h_{ij} = 1$; $h_{ij} = 0$ otherwise.

Traditionally, an ensemble of LDPC codes is characterized by its *variable degree distribution* $\Lambda(x) = \sum_{i=1}^{d_v^{max}} \lambda_i x^{i-1}$ and its *check degree distribution* $\Phi(x) = \sum_{i=1}^{d_c^{max}} \phi_i x^{i-1}$, where d_v^{max} is the maximum variable degree, and d_c^{max} is the maximum check degree. This couple of degree distributions verify $\sum_{i=1}^{d_v^{max}} \lambda_i = 1$, and $\sum_{i=1}^{d_c^{max}} \phi_i = 1$. Each variable node x has degree d_v, meaning that it is connected to exactly d_v check nodes;

similarly, each check node s has degree d_c, meaning that its connected to exactly d_c variable nodes. $\forall i \in [d_v^{min}, d_v^{max}]$, λ_i is the proportion of variable nodes having degree i; similarly, $\forall i \in [d_c^{min}, d_c^{max}]$, ϕ_i is the proportion of check nodes having degree i.

A *subgraph* of G is a graph whose nodes and edges sets are subsets of those of G. For a given variable node x_j, we define its *neighborhood within depth* l, noted $\mathcal{N}_{x_j}^l$, as the set formed by all the check nodes reached by a subgraph spreading from x_j within depth l. Its complementary set is noted $\overline{\mathcal{N}}_{x_j}^l$.

The *girth* of x_j is the minimum depth at which a subgraph spreading from x_j loops back to x_j. The girth of the code (or of the graph) is the minimum girth among all its variable nodes.

B.1.2 Presentation of the algorithm

The PEG algorithm is a suboptimal algorithm to construct a Tanner graph with a relatively large girth. The local girth of each variable node is maximized whenever a new edge is added to the node. The resulting graph grows in an edge-by-edge manner, optimizing each local girth.

The principle of the algorithm is to find the most distant check node from the current variable, in terms of the subgraph depth, and then to connect the two nodes together. This principle is described in more details in the sequel. Whenever a subgraph from symbol node x_j is expanded before an edge is established, two situations can occur:

1. The cardinality of $\mathcal{N}_{x_j}^l$ stops increasing but is smaller than $(N-K)$; not all check nodes are reachable from x_j, so the PEG algorithm chooses the check node to connect to among the ones that are *not* reachable, thus not creating any additional cycle. This often occurs in the initial phase of graph construction.

2. $\mathcal{N}_{x_j}^l \neq \emptyset$, but $\mathcal{N}_{x_j}^{l+1} = \emptyset$; all check nodes are reachable from x_j, so the algorithm chooses the one that is at the largest distance from x_j, say at depth $(l+1)$, so that the cycle created by establishing such an edge is of the largest possible length $2(l+2)$.

The Progressive Edge Growth algorithm can be summarized as follows:

for $j = 0$ to $(N-1)$ **do**
  for $k = 0$ to $d_v^j - 1$ **do**
    if $k = 0$ **then**
      Find s_i, a check node which has the lowest check-node degree under the current graph setting (respecting the check degree distribution).
      Establish the first edge incident to x_j: $E_{s_j}^0 \leftarrow edge(s_i, x_j)$.
    else
      Expand a subgraph from x_j, up to the depth l, under the current graph setting, s.t. the cardinality of $\mathcal{N}_{x_j}^l$ stops increasing, but is less than $(N-K)$, or $\mathcal{N}_{x_j}^l \neq \emptyset$ but $\mathcal{N}_{x_j}^{l+1} = \emptyset$.
      Pick s_i from $\mathcal{N}_{x_j}^l$, a check node that has the lowest check-node degree (respecting the check degree distribution).

Establish the k-th edge incident to x_j: $E_{s_j}^k \leftarrow edge(s_i, x_j)$.
end if
end for
end for

Me may face a situation in which multiple choices exist because multiple check nodes in $\mathcal{N}_{x_j}^l$ might have the same lowest degree, particularly in the initial phase of PEG construction. There are two main approaches to solve this problem. The first one is to randomly select one of these check nodes; the second is to always select one according to its position in an arbitrary sorted order. In our implementation, we decide to randomly pick the check nodes.

B.2 Proposed PEG suited to non-asymmetric SW problem

We use the same notation as for the original PEG. Our task of constructing a good DSC LDPC code is coupled with the aim to construct an optimal code for the non-asymmetric SW problem. As Lemma 4.1 imposes a triangular $\mathbf{B}$ part to the matrix $\mathbf{H}$, we first explain how to match that condition.

B.2.1 Construction of the required triangular part

This triangular part is of size $(N-K) \times (N-K)$. First, we work on the matrix representation and fill the $\mathbf{B}$ part with two diagonal lines of 1's, *s.t.* the $(N-K-1)$ first variable nodes corresponding to the square matrix have degree *two*, and the last variable node has degree *one*. If $\lambda_2 \geq \frac{N-K}{N}$, then the matrix $\mathbf{B}$ is set. Otherwise, suppose that we have $\lambda_2 + \lambda_3 \geq \frac{N-K}{N}$. Then we fill the $(N-K) - \lambda_2 N$ first columns of $\mathbf{B}$ with a diagonal line of 1's, s.t. the corresponding variable nodes have degree *three*. If $\lambda_2 + \lambda_3 < \frac{N-K}{N}$, then we complete $\mathbf{B}$ with degree *four* columns taking into account λ_4. However, in practice, there should be no need to go further than λ_3.

As an example, consider the rate-$\frac{1}{2}$ LDPC code of variable degree distribution $\Lambda(x) = 0.483949x + 0.294428x^2 + 0.085134x^5 + 0.074055x^6 + 0.062433x^{19}$ (used in Section 4.3.3.4). As $\Lambda_2 + \Lambda_3 = 0.7784 \geq 0.5$, the proportion of variable nodes of degree two and three is large enough to form the $\mathbf{B}$ part. Practically, for a 1000-long code, there are 484 columns with *two* 1's and 16 columns with *three* 1's forming the $\mathbf{B}$ part (see Fig. B.1). The remaining columns of the matrix include 278 columns with *three* 1's, 85 columns with *six*, 75 columns with *seven* and 62 columns with *twenty*.

Now that the $\mathbf{B}$ part of the matrix is set, we build the graph G taking into account that the variable node x_j is connected the check node s_i in the graph if, and only if, $h_{ij} = 1$ in the matrix.

B.2.2 Building the graph with complementary edges

Once the edges corresponding to the correct $\mathbf{B}$ part of the matrix are set, we apply the PEG algorithm, making sure that the existing edges are never modified. Only

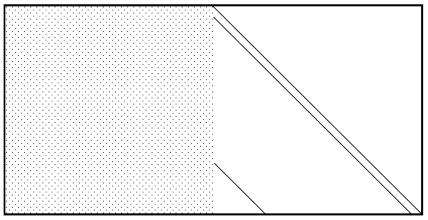

Figure B.1: Shape of the designed LDPC code's parity-check matrix.

the $(N-K)$ variable nodes corresponding to the columns of $\mathbf{H}$ *not* part of $\mathbf{B}$ need to be processed. The new algorithm is summarized as follows:

for $j = 0$ to $(N-K-1)$ **do**
  for $k = 0$ to $d_v^j - 1$ **do**
    if $k = 0$ **then**
      Find s_i, a check node which has the lowest check-node degree under the current graph setting (respecting the check degree distribution).
      Connect the first edge incident to x_j: $E_{s_j}^0 \leftarrow edge(s_i, x_j)$.
    else
      Expand a subgraph from x_j, up to the depth l, under the current graph setting, s.t. the cardinality of $\mathcal{N}_{x_j}^l$ stops increasing, but is less than $(N-K)$, or $\mathcal{N}_{x_j}^l \neq \emptyset$ but $\mathcal{N}_{x_j}^{l+1} = \emptyset$.
      Pick s_i from $\mathcal{N}_{x_j}^l$, a check node that has the lowest check-node degree (respecting the check degree distribution).
      Connect the k-th edge incident to x_j: $E_{s_j}^k \leftarrow edge(s_i, x_j)$.
    end if
  end for
end for

As we impose a *deterministic* structure for the part $\mathbf{B}$ of the matrix, some variable nodes may not be connected to enough check nodes at the end of the algorithm, with respect to their theoretical degree. In such case, it might be preferable to leave the degree distribution unmatched, instead of adding extra edges that could deteriorate the performance of the code, by decreasing the global girth of the graph.

Note that the only degree-*one* variable node, imposed by the construction of the part $\mathbf{B}$ leaves the graph with a sub-optimality that we must accept, as part of the trade-off between optimality for DSC and for the non-asymmetric problem. If needed, the source bits corresponding to this position can be sent directly to the decoder as systematic bit.

Appendix C

The five video sequences

These pictures are shown in Fig. C.1 to help the reader have a visual on each video sequence. Since we improve their compression rates, but we do not improve their quality, in all the contributions of this Thesis, we do not show them after each test of Chapter 5.

The sequences are *Hall Monitor*, *Foreman* (with the Siemens logo), *CoastGuard*, *Flower*, and *Soccer*. They have QCIF spatial resolution (176×144 pixels), 15Hz temporal resolution, which means 7.5Hz for the WZ frames when a GOP size of 2 is used (which we choose for all our tests).

Figure C.1: A visual of the five video sequences. From top to bottom, and from left to right: *Hall Monitor*, *Foreman* (with the Siemens logo), *CoastGuard*, *Flower*, and *Soccer*.

Bibliography

[AAD+07] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret. The DISCOVER codec: Architecture, techniques and evaluation. In *Proceedings of Picture Coding Symposium (PCS)*, November 2007.

[ABP06] J. Ascenso, C. Brites, and F. Pereira. Content adaptive Wyner-Ziv video coding driven by motion activity. In *IEEE International Conference on Image Processing (ICIP)*, pages 605–608, October 2006.

[AG02] A. Aaron and B. Girod. Compression with side information using turbo codes. In *Data Compression Conference*, pages 252–261, April 2002.

[AL01] D. Arnold and H.-A. Loeliger. On the information rate of binary input channels with memory. *IEEE International Conference on Communications*, 9:2692–2695, June 2001.

[AMGT07] X. Artigas, S. Malinowski, C. Guillemot, and L. Torres. Overlapped quasi-arithmetic codes for distributed video coding. *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pages 9–12, September 2007.

[AMGT08] X. Artigas, S. Malinowski, C. Guillemot, and L. Torres. Overlapped arithmetic codes with memory. *Proceedings of the 16th European Signal Processing Conference (EUSIPCO)*, August 2008.

[AZG02] A. Aaron, Rui Zhang, and B. Girod. Wyner-ziv coding of motion video. *Conference Record of the Thirty-Sixth Signals, Systems and Computers*, 1:240–244, November 2002.

[BCJR74] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, 20(2):284–287, March 1974.

[Ber72] T. Berger. *Rate Distortion theory: a mathematical basis for data compression.* Prentice-Hall, Inc., January 1972.

[BG96] C. Berrou and A Glavieux. Near optimum error correcting coding and decoding: Turbo codes. *IEEE Transactions on Communications*, 44(10):1261–1271, October 1996.

[BKW08] F. Bassi, M. Kieffer, and C. Weidmann. Source coding with intermittent and degraded side information at the decoder. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2941–2944, March 2008.

[BM01] J. Bajcsy and P. Mitran. Coding for the Slepian-Wolf problem with turbo codes. In *Proceedings of Globecom*, volume 2, pages 1400–1404, November 2001.

[Cov75] T. Cover. A proof of the data compression theorem of Slepian-Wolf for ergodic sources. *IEEE Transactions on Information Theory*, 22(2):226–268, March 1975.

[CT91] T. M. Cover and J. M. Thomas. *Elements of Information Theory*. Wiley, New–York, 1991.

[CWC09] S. Cheng, S. Wang, and L. Cui. Adaptive Slepian-Wolf decoding using particle filtering based belief propagation. In *Proceedings of the 47th annual Allerton conference on Communication, control, and computing*, pages 607–612, September 2009.

[dis] DISCOVER project web page. http://www.discoverdvc.org/.

[DLV02] A. Dieter, H.-A Loeliger, and P. O. Vontobel. Computation of information rates from finite-state source/channel models. *40th Annual Allerton Conference on Communication, Control, and Computing*, October 2002.

[Eck04] A. W. Eckford. *Low-density parity-check codes for Gilbert-Elliott and Markov-modulated channels*. PhD thesis, Thesis for the Degree of Doctor of Philosophy, University of Toronto, Canada, 2004.

[Eck05] A. W. Eckford. Analysis of low-density parity-check codes for the Gilbert-Elliott channel. *IEEE Transactions on Information Theory*, 51(11):3872–3889, November 2005.

[Ell63] E. O. Elliott. Estimates of error rates for codes on burst-noise channels. *Bell-System Technical Journal*, 42:1977–1997, September 1963.

[FJ09] Y. Fang and J. Jeong. Correlation parameter estimation for LDPC-based Slepian-Wolf coding. *IEEE Communications Letters*, 13(1):37–39, January 2009.

[Gal62] R. Gallager. Low-Density Parity-Check Codes. *IEEE Transactions on Information Theory*, 8(1):21–28, 1962.

[GD05] N. Gehrig and P. L. Dragotti. Symmetric and asymmetric Slepian-Wolf codes with systematic and non-systematic linear codes. *IEEE communications letters*, 9(1):61–63, January 2005.

[GF04] J. Garcia-Frias. Decoding of low-density parity-check codes over finite-state binary Markov channels. *IEEE Transactions on Communications*, 52(11):1840–1843, November 2004.

[GFC04] J. Garcia-Frias and F. Cabarcas. Approaching the slepian-wolf boundary using practical channel codes. In *Proceedings of the International Symposium on Information Theory (ISIT)*, page 330, June 2004.

[GFV97] J. Garcia-Frias and J.D. Villasenor. Combining hidden markov source models and parallel concatenated codes. *IEEE Communications Letters*, 1(4):111–113, July 1997.

[GFV02] J. Garcia-Frias and J.D. Villasenor. Turbo decoding of gilbert-elliot channels. *IEEE Transactions on Communications*, 50(3):357–363, March 2002.

[GFZ01] J. Garcia-Frias and Y. Zhao. Compression of correlated binary sources using turbo codes. *IEEE Communications Letters*, 5:417–419, October 2001.

[Gil60] E. N. Gilbert. Capacity of a burst-noise channel. *Bell-System Technical Journal*, pages 1253–1265, March 1960.

[GMO07] M. Grangetto, E. Magli, and G. Olmo. Distributed arithmetic coding. *IEEE Communications Letters*, 11(11):883–885, November 2007.

[GMO09] M. Grangetto, E. Magli, and G. Olmo. Distributed arithmetic coding for the Slepian-Wolf problem. *IEEE Transactions on Signal Processing*, 57(6):2245–2257, June 2009.

[GMTO08] M. Grangetto, E. Magli, R. Tron, and G. Olmo. Rate-compatible distributed arithmetic coding. *IEEE Communications Letters*, 12(8):575–577, August 2008.

[HEA05] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold. Regular and irregular progressive edge-growth Tanner graphs. *IEEE Transactions on Information Theory*, 51(1):386–398, January 2005.

[HTZR09] C. Herzet, V. Toto-Zarasoa, and A. Roumy. Error resilient non-asymmetric Slepian-Wolf coding. In *IEEE International Conference on Communications (ICC)*, pages 1–5, June 2009.

[JKM00] H. Jin, D. Khandekar, and R. J. McEliee. Irregular repeat-accumulate codes. *International symposium on turbo codes (ISTC)*, pages 1–8, September 2000.

[JW05] S. J. Johnson and S. R. Weller. Constructions for irregular repeat-accumulate codes. In *IEEE International Symposium on Information Theory (ISIT)*, pages 179–183, September 2005.

[Kay93] S. M. Kay. *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Inc., 1993.

[KB82] A.H. Kaspi and T. Berger. Rate-distortion for correlated sources with partially separated encoders. *IEEE Transactions on Information Theory*, 28(6):828–840, November 1982.

[KFL01] F. R. Kschischang, B. J. Frey, and H. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001.

[KNG07] D. Kubasov, J. Nayak, and C. Guillemot. Optimal reconstruction in Wyner-Ziv video coding with multiple side. In *International Workshop on Multimedia Signal Processing (MMSP)*, pages 183–186, October 2007.

[KTRR03] P. Koulgi, E. Tuncel, S.L. Regunathan, and K. Rose. On zero-error source coding with decoder side information. *IEEE Transactions on Information Theory*, 49(1):99–111, January 2003.

[Kub08] D. Kubasov. *Codage de sources distribuées :Nouveaux outils et applications à la compression vidéo*. PhD thesis, Thèse de doctorat en Informatique, Université de Rennes 1, France, December 2008.

[LA05] J. Li and H. Alqamzi. An optimal distributed and adaptive source coding strategy using rate-compatible punctured convolutional codes. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3:685–688, March 2005.

[Laj06] K. Lajnef. *Etude du codage de sources distribuées pour de nouveaux concepts en compression vidéo*. PhD thesis, Thèse de doctorat en Informatique, Université de Rennes 1, France, 2006.

[ldp] LDPCOpt. http://ipgdemos.epfl.ch/ldpcopt/.

[LGS04] K. Lajnef, C. Guillemot, and P. Siohan. Distributed coding of three sources using punctured turbo codes. In *IEEE International workshop on Multimedia Signal Processing (MMSP)*, pages 307–310, September 2004.

[LGS06] K. Lajnef, C. Guillemot, and P. Siohan. Distributed coding of three binary and gaussian correlated sources using punctured turbo codes. *Signal Processing*, 86(11):3131–3149, 2006.

[LVG07a] Y. Lin, D. Varodayan, and B. Girod. Image authentication and tampering localization using distributed source coding. In *IEEE Multimedia Signal Processing Workshop (MMSP)*, pages 393–396, October 2007.

[LVG07b] Y. Lin, D. Varodayan, and B. Girod. Image authentication based on distributed source coding. In *IEEE International Conference on Image Processing (ICIP)*, pages 5–8, September 2007.

[LXG02] A. D. Liveris, Z. Xiong, , and C. N. Georghiades. Compression of binary sources with side information at the decoder using LDPC codes. *IEEE Communications Letters*, 6(10):440–442, October 2002.

[MBD89] M. Mushkin and I. Bar-David. Capacity and coding for the Gilbert-Elliot channels. *IEEE Transactions on Information Theory*, 35(6):1277–1290, November 1989.

[PCR03] S. Pradhan, J. Chou, and K. Ramchandran. Duality between source coding and channel coding and its extension to the side information case. *IEEE Transactions on Information Theory*, 49(5):1181–1203, May 2003.

[PR99] S. S. Pradhan and K. Ramchandran. Distributed source coding using syndromes (DISCUS): Design and construction. *Proceedings of the IEEE Data Compression Conference (DCC)*, pages 158–167, March 1999.

[PR00] S. S. Pradhan and K. Ramchandran. Distributed source coding: symmetric rates and applications to sensor networks. In *Proceedings of the IEEE Data Compression Conference*, pages 363–372, March 2000.

[PR02] R. Puri and K. Ramchandran. PRISM: A new robust video coding architecture based on distributed compression principles. *Proceedings of the 40th Allerton Conference on Communication, Control and Computing*, October 2002.

[PSM98] K. Podgorski, G. Simons, and Y. Ma. On estimation for a binary symmetric channel. *IEEE transactions on information theory*, 44(3):1261–1272, May 1998.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[RAG04] S. Rane, A. Aaron, and B. Girod. Systematic lossy forward error protection for error-resilient digital video broadcasting - a Wyner-Ziv coding approach. In *IEEE International Conference on Image Processing*, volume 5, pages 3101–3104, October 2004.

[RBG08] S. Rane, P. Baccichet, and B. Girod. Systematic lossy error protection of video signals. *IEEE transactions on Circuits and Systems for Video Technology*, 18(10):1347–1360, October 2008.

[Rez05] M. Rezaeian. Computation of capacity for Gilbert-Elliott channels, using a statistical method. *6th Australian Communications Theory Workshop*, pages 56–61, February 2005.

[RG07] A. Roumy and D Gesbert. Optimal matching in wireless sensor networks. *IEEE Journal on Selected Topics in Signal Processing, Special issue on Convex Optimization Methods in Signal Processing*, 1(4):725–735, December 2007.

[RLG07] A. Roumy, K. Lajnef, and C. Guillemot. Rate-adaptive turbo-syndrome scheme for Slepian-Wolf coding. In *41st Asilomar Conference on Signals, Systems and Computers*, pages 545–549, November 2007.

[RSU01] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, February 2001.

[SF05] M. Sartipi and F. Fekri. Distributed source coding in wireless sensor networks using LDPC coding: The entire Slepian-Wolf rate region. *IEEE Wireless Communications and Networking Conference*, 4:1939–1944, March 2005.

[Sha59] C. E. Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE National Convention Record*, pages 142–163, 1959.

[Sim91] G. Simons. Estimating distortion in a binary symmetric channel consistently. *IEEE transactions on information theory*, 37(5):1466–14700, September 1991.

[SLXG04] V. Stankovic, A. D. Liveris, Z. Xiong, and C. N. Georghiades. Design of Slepian-Wolf codes by channel code partitioning. In *Data Compression Conference (DCC)*, pages 302–311, March 2004.

[SW73] D. Slepian and J. K. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 19(4):471–480, July 1973.

[Tan81] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, September 1981.

[TL05a] P. Tan and J. Li. Enhancing the robustness of distributed compression using ideas from channel coding. In *IEEE Global Telecommunications Conference (GLOBECOM)*, volume 4, pages 2385–2389, December 2005.

[TL05b] P. Tan and J. Li. A practical and optimal symmetric Slepian-Wolf compression strategy using syndrome formers and inverse syndrome formers. In *Proceedings of 43rd Annual Allerton Conference on Communication, Control and Computing*, September 2005.

[TZMRO09] V. Toto-Zarasoa, E. Magli, A. Roumy, and G. Olmo. On distributed arithmetic codes and syndrome based turbo codes for Slepian-Wolf coding of non uniform sources. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 2249–2253, August 2009.

[VAG06] D. Varodayan, A. Aaron, and B. Girod. Rate-adaptive codes for distributed source coding. *European Association for Signal Processing (EURASIP)*, 86(11):3123–3130, November 2006.

[Vit67] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[VLG08] D. Varodayan, Y. Lin, and B. Girod. Audio authentication based on distributed source coding. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 225–228, March 2008.

[Wu83] C. F. J. Wu. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11(1):95–103, 1983.

[Wyn74] A. Wyner. Recent results in the Shannon theory. *IEEE Transactions on Information Theory*, 20(1):2–10, January 1974.

[Wyn78] A. D. Wyner. The rate-distortion function for source coding with side information at the decoder-II: general sources. *Information and Control*, 38(1):60–80, July 1978.

[WZ76] A. Wyner and J. Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 21(1):1–10, January 1976.

[ZE03] Q. Zhao and M. Effros. Lossless and near-lossless source coding for multiple access networks. *IEEE Transactions on Information Theory*, 49(1):112–128, January 2003.

[ZRS07] A. Zia, J. P. Reilly, and S. Shirani. Distributed parameter estimation with side information: A factor graph approach. In *Proceedings of the IEEE International Symphosium on Information Theory (ISIT)*, pages 2556–2560, June 2007.

Publications

International conferences

[1] Velotiaray Toto-Zarasoa, Aline Roumy, Christine Guillemot - **Rate-adaptive codes for the entire Slepian-Wolf region and arbitrarily correlated sources** - *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* - Pages 2965–2968, April, 2008.

[2] Velotiaray Toto-Zarasoa, Aline Roumy, Christine Guillemot, Cédric Herzet - **Robust and fast non asymmetric Distributed Source Coding using Turbo Codes on the Syndrome trellis** - *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* - Pages 2477–2480, April, 2009.

[3] Cédric Herzet, Velotiaray Toto-Zarasoa, Aline Roumy - **Error resilient non-asymmetric Slepian-Wolf Coding** - *IEEE International Conference on Communications (ICC)* - Pages 1–5, June, 2009.

[4] Velotiaray Toto-Zarasoa, Enrico Magli, Aline Roumy, Gabriella Olmo - **On Distributed Arithmetic Codes and Syndrome Based Turbo Codes for Slepian-Wolf Coding of Non Uniform Sources** - *European Signal Processing Conference (EUSIPCO)* - Pages 2249–2253, August, 2009.

[5] Velotiaray Toto-Zarasoa, Aline Roumy, Christine Guillemot - **Non-uniform source modeling for Distributed Video Coding** - *Accepted for European Signal Processing Conference (EUSIPCO)* - Pages 1889–1893, August, 2010.

[6] Velotiaray Toto-Zarasoa, Aline Roumy, Christine Guillemot - **Non-asymmetric Slepian-Wolf coding of non-uniform Bernoulli sources** - *6th International Symposium on Turbo codes (ISTC)* - Pages 314–318, September, 2010.

[7] Velotiaray Toto-Zarasoa, Aline Roumy, Christine Guillemot - **Hidden Markov model for Distributed Video Coding** - *Accepted for IEEE International Conference on Image Processing (ICIP)* - Pages 3709–3712, September, 2010.

Journals

[8] Velotiaray Toto-Zarasoa, Aline Roumy, Christine Guillemot - **Source modeling for Distributed Video Coding** - *Submitted to IEEE Transactions on Image Processing* - 2010.

[9] Velotiaray Toto-Zarasoa, Aline Roumy, Christine Guillemot - **Maximum Likelihood BSC parameter estimation for the Slepian-Wolf problem** - *Submitted to IEEE Communications Letter* - 2010.

List of Figures

Résumé

Le *codage de sources distribuées* est une technique permettant de compresser plusieurs sources corrélées sans aucune coopération entre les encodeurs, et sans perte de débit si leur décodage s'effectue conjointement. Fort de ce principe, le *codage de vidéo distribué* exploite la corrélation entre les images successives d'une vidéo, en simplifiant au maximum l'encodeur et en laissant le décodeur exploiter la corrélation.

Parmi les contributions de cette thèse, nous nous intéressons dans une première partie au codage *asymétrique* de sources binaires dont la distribution *n'est pas uniforme*, puis au codage des *sources à états de Markov cachés*. Nous montrons d'abord que, pour ces deux types de sources, exploiter la distribution au décodeur permet d'augmenter le taux de compression. En ce qui concerne le canal binaire symétrique modélisant la corrélation entre les sources, nous proposons un outil, basé sur l'algorithme EM, pour en estimer le paramètre. Nous montrons que cet outil permet d'obtenir une estimation rapide du paramètre, tout en assurant une précision proche de la borne de Cramer-Rao.

Dans une deuxième partie, nous développons des outils permettant de décoder avec succès les sources précédemment étudiées. Pour cela, nous utilisons des codes Turbo et LDPC basés syndrome, ainsi que l'algorithme EM. Cette partie a été l'occasion de développer des nouveaux outils pour atteindre les bornes des codages *asymétrique* et *non-asymétrique*. Nous montrons aussi que, pour les sources non-uniformes, le rôle des sources corrélées n'est pas symétrique.

Enfin, nous montrons que les modèles de sources proposés modélisent bien les distributions des plans de bits des vidéos; nous montrons des résultats prouvant l'efficacité des outils développés. Ces derniers permettent d'améliorer de façon notable la performance débit-distorsion d'un codeur vidéo distribué, mais sous certaines conditions d'*additivité* du canal de corrélation.

Abstract

Distributed source coding is a technique that allows to compress several correlated sources, without any cooperation between the encoders, and without rate loss provided that the decoding is joint. Motivated by this principle, *distributed video coding* has emerged, exploiting the correlation between the consecutive video frames, tremendously simplifying the encoder, and leaving the task of exploiting the correlation to the decoder.

The first part of our contributions in this thesis presents the asymmetric coding of binary sources that are not uniform. We analyze the coding of *non-uniform* Bernoulli sources, and that of *hidden Markov sources*. For both sources, we first show that exploiting the distribution at the decoder clearly increases the decoding capabilities of a given channel code. For the binary symmetric channel modeling the correlation between the sources, we propose a tool to estimate its parameter, thanks to an EM algorithm. We show that this tool allows to obtain fast estimation of the parameter, while having a precision that is close to the Cramer-Rao lower bound.

In the second part, we develop some tools that facilitate the coding of the previous sources. This is done by the use of syndrome-based Turbo and LDPC codes, and the EM algorithm. This part also presents new tools that we have developed to achieve the bounds of *asymmetric* and *non-asymmetric* distributed source coding. We also show that, when it comes to non-uniform sources, the roles of the correlated sources are not symmetric.

Finally, we show that the proposed source models are well suited for the video bit planes distributions, and we present results that proof the efficiency of the developed tools. The latter tools improve the rate-distortion performance of the video codec in an interesting amount, provided that the correlation channel is *additive*.